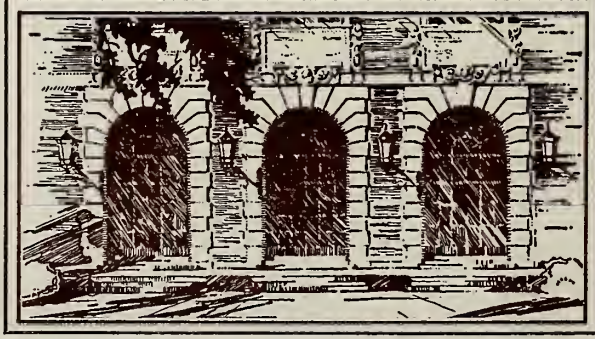


UIUCDCS-R-72-530

IMPLEMENTATION OF BASIC SOFTWARE FOR SIGNIFICANT DIGIT ARITHMETIC

by
Steven See Sun Lai

June, 1972

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

IMPLEMENTATION OF BASIC SOFTWARE FOR SIGNIFICANT DIGIT ARITHMETIC

BY

STEVEN SEE SUN LAI

1972

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

This work was supported in part by the National Science Foundation
Grant No. US NSF-GJ-328.

ACKNOWLEDGMENT

The author wishes to thank his advisor, Professor J. R. Phillips, for his suggestion of this thesis topic, his subsequent guidance, and his patience throughout its preparation. Special thanks go to Bob Bloemer of the OL/2 implementation group for his many suggestions and stimulating discussions which greatly contributed to this effort.

Furthermore, the author is indebted to the Office of University Administrative Data Processing for their financial support and to the Department of Computer Science for the computing facilities necessary for this work to be completed.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Automatic error control has been studied and implemented since the late fifties (15). Over the past decade refinements have been made and further analytic results have come to light (4, 5, 8, 9, 16). It appears that the state of the art suggests that designers and implementers should try to embed these rules into their software systems. This provided the motivation for this thesis, which was to implement the rules of "unnormalized significant digit arithmetic" of Metropolis (15, 9) and Ashenhurst (2, 4) and embed them in the array language OL/2 (20). However, these software packages are also adaptable to other compatible software systems, since they are written in assembly language for the IBM 360/370 machines.

Numerical errors in digital computation arise from three principle sources: "inherent" errors which result from inaccurate data, "analytic" errors which result from replacing an infinite process with a finite process, and "generated" errors which result from using finite-precision arithmetic (1). Often, computed results are reported without specifying their accuracy because inherent and generated errors have been treated rather lightly. Presently, most floating-point arithmetic is performed in normalized form which leads to the retention of nonsignificant digits in final computed results. There is no control of

significant digits when experimental data is used as input. Even with precise input, ill-conditioning or instability in the programmed algorithm may cause erosion of precision, thus a specification of error or significance should accompany output numbers.

As a consequence of its practical importance, analytical investigations into, as well as implementation of, significance control has been made by several earlier investigators on several different machines. Notable among these are Gardiner and Metropolis (10), Goldstein (11), Ashenhurst (3), and Bright (7). Each agree to the two rules of significant digit arithmetic: (1) for addition and subtraction, the resulting exponent is equal to the larger of the input exponents, and (2) for multiplication and division, the number of significant digits in the result is equal to that of the less significant operand. They differ, however, in their implementations of the internal representation of significant digit numbers. Because each machine employed a different exponent base, corresponding significant adjustment rules for library functions also reflect this difference.

All numerical quantities may be regarded as either exact or approximate. However, the finite presentation of number in computer requires a further characterization. If an exact number can be represented in some preassigned number of digits (the computer word length), then it is said to be <u>precise</u>. Other exact numbers belong to the approximate class are called

imprecise. Using these two categories and base of two, Gardiner and Metropolis (9) introduced the ABC (analyzed binary computation) form of arithmetic. The goals of ABC arithmetic are: (1) to determine if the computed result is precise or unprecise, and (2) to specify the number of significant digits in the imprecise result. Thus, ABC arithmetic uses the rules of significant digit arithmetic and defines an additional set of rules for combining both precise and imprecise numbers. Whenever we use the term "significant digit arithmetic" we mean to include the new rules of ABC arithmetic as well. The present work follows the approach of Gardiner and Metropolis (9), except that the base two rules must be replaced by base sixteen rules, since the implementation is for the IBM 360/370 machines.

Overview of the Thesis

The next chapter defines the basic rules for combining precise and imprecise numbers in the context of ABC arithmetic. Two kinds of zeroes, true and relative, are also discussed in this context, along with the internal format for imprecise and precise numbers.

Chapter 3 discusses the significant digit adjustment rules for elementary functions. In particular, Richtmyer's adjustment rule for functions is derived. The discussion follows the rules that were prescribed by Ashenhurst (4), with suitable changes for base sixteen rather than base two numbers.

The fourth chapter explains the format of input/output for precise and imprecise numbers. The significance number base conversion algorithms of Kanner (13) will be presented.

Chapter 5 briefly explores some extensions to the present work, and also some of the problems which the uninitiated user must be aware of when using significance arithmetic.

CHAPTER 2

PRECISE/IMPRECISE ARITHMETIC

2.1 Unnormalized Floating Point Numbers

A floating point number may be represented by a pair (e, f), where e is the integral exponent and f the real valued fraction part. Most of today's computers use normalized arithmetic in which the floating point number has its non-zero fractional part satisfying $r^{-1} \leq f < r^{0}$, where r is the number base. This format alone cannot specify the number of significant digits in the fraction. By unnormalizing the number, it is possible to convey the significant digits in the fraction, since $(e + m, r^{-m}f)$ represents the same number as (e, f) for any integer m. The integer m describes the number of leading zeroes in the fraction. By definition, therefore, imprecise numbers are represented internally as unnormalized numbers with at least one leading zero. Precise numbers, on the other hand, are represented internally as normalized numbers. This distinction between precise and imprecise numbers forms the basis for the rules of ABC arithmetic (9) which are discussed subsequently.

2.2 True Zero and Relative Zero

In significant digit arithmetic there are two kinds of zeroes: relative zero, which has no significant digits and true zero, which has an infinite number. Both zeroes have zero fractional parts, and obviously, there is no sign attached to the

fractions. True zero is designated by using the smallest machine representable exponent, in general, this is just the usual machine representation of zero. Relative zero, on the other hand, may have any exponent value. Clearly, the rules for operating with these two zeroes is quite different. True zero is a precise number in the sense that it acts like zero. In contrast, relative zero is a degenerate imprecise number with no significant digits but with an exponent field that indicates an "order of magnitude." Since relative zero contains some information about a number besides complete loss of significance, it seems appropriate to consider separate rules for operating with it. It should be clear that a relative zero may occur during a computational process as the result of very basic operations. For instance, a relative zero may occur when two nearly equal imprecise numbers are subtracted. If the original operands had a large exponent, then the relative zero has a large exponent. This indicates that a relative zero is not necessarily close to zero, but is merely a bound on the result.

2.3 Internal Representation of Numbers

The present work was implemented on a IBM 360/75 computer using long floating point numbers. These numbers use 64 bits: 7 bits for the base 16 exponent and 57 bits for the fraction. (See Figure 1.) The zero-th bit is used for the sign of the fraction. The exponent is represented in excess-64 code with limits of -63 and +64. True zero has an exponent of -64 and has zeroes in all 64 bits. (This is the standard floating point

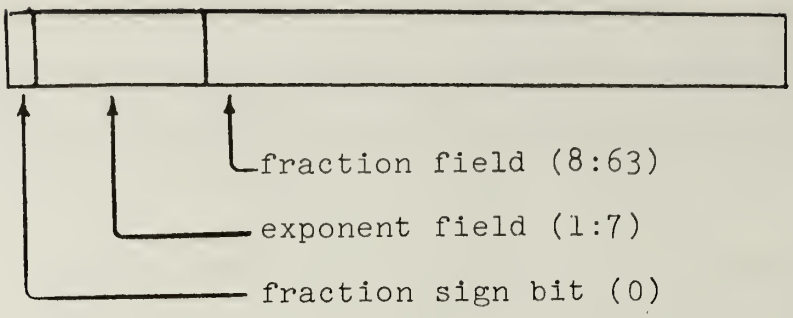
zero.) Relative zero is represented by a zero fraction, a zero in the sign position, and arbitrary exponent e in the range $-63 \leq e \leq 63$. The sign of relative zero is neither positive or negative.

A precise number is represented as normalized floating point number with 14 hexadecimal digits in the fractional part. Only precise numbers are normalized. The range of IBM's long floating point number is from 10^{-77} to 10^{+75}.

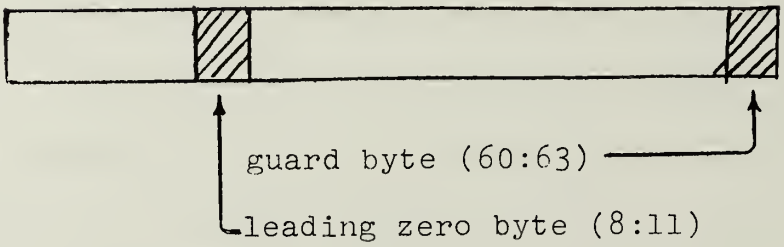
Since an imprecise number is an approximate quantity, it is represented with fewer fractional digits. The first 4-bit fractional byte is always zero and the last hexadecimal digit is always defined to be a nonsignificant guard byte. Thus, the imprecise fractional field uses the middle 12 hexadecimal digits. Because imprecise numbers are unnormalized, they can only represent numbers between 10^{-77} and 10^{+49}. The guard byte is used in input/output conversion and also serves as a guard against round-off errors. Kanner (13) points out that if input conversion followed by output reconversion is to be an identity operation at least one guard bit must be kept. Because the exponent base is sixteen, it is necessary in this implementation to use an entire byte of 4 bits.

2.4 Base Operations

In this section the operations of addition (including subtraction), multiplication, and division in ABC arithmetic are described. If one of the operands is imprecise, then the result is imprecise or a relative zero. The significant digit rules are



a) IBM 360/370 long floating point format



b) Imprecise number

Figure 1. Internal number format

used for imprecise numbers. Let $X_1 = (e_1, f_1)$ and $X_2 = (e_2, f_2)$ be the operands, and $X_3 = (e_3, f_3)$ the results, then the rules of significant digit arithmetic are defined as follows:

1. For addition or subtraction $e_3 = \max \{e_1, e_2\}$,
2. For multiplication or division $m_3 = \max \{m_1, m_2\}$

where m is the number of leading zeroes in the fraction. If a zero fraction results, the exponent of the relative zero is fixed according to the rules of regular arithmetic. That is, in addition e_3 is the maximum exponent of operands, in multiplication e_3 is the algebraic sum of the exponents, and in division e_3 is the difference between the exponent of the numerator and the exponent of the denominator. There are some obvious problems with using the 360/370 floating point arithmetic. If the unnormalization of imprecise numbers is not possible, then the ABC arithmetic routines signal exponent overflow. This is why imprecise numbers have an upper limit of 10^{+49} instead of the 10^{+75} of the precise number.

A necessary but not sufficient condition for a precise result is that both operands are precise. The result would be imprecise if it cannot be represented exactly in the long floating point word. For multiplication and division, the low order part of the immediate product is generated and examined for any non-zero bits. (The 360/370 machines generate only the higher order part.) Since the product is almost twice the length of input operand, a quick test for a precise multiply is to check if both precise inputs are contained in the short floating point

number format which has 32 bits and 6 hexadecimal fractional digits. For addition, the precise check is to subtract the result by the operand with the larger exponent. This subtraction would have no shifting of fraction to align the exponents The result is then compared with the other input operand. If they are not equal, fractional digits were lost in the addition; so the result is imprecise. The special case of a true zero resulting from the subtraction of the precise number by itself is checked by comparing the input fractional digits for equality.

As noted earlier, true zero is the zero element in ABC arithmetic. In addition, it is the identity element. A true zero product results if one of the operands is a true zero. Division by true zero is signaled as a divide exception and a true zero dividend results in a true zero quotient.

On the other hand, the rules for operating with relative zero are very different. If both operands are relative zeroes, then the exponent of the resulting relative zero exponent must be adjusted. In extreme cases, an exponent overflow in a relative zero raises an error condition while an exponent underflow is replaced with the smallest relative zero. In multiplication and division, the result is generally a relative zero if one of the operands is a relative zero. It is interesting to note that the quotient of a non-zero divided by a relative zero is defined as a relative zero. In this case, multiplication of quotient and divisor does not give the dividend.

Table 1

ABC Multiplication

	O_T	O_R	X_I	X_P
O_T	O_T	O_T	O_T	O_T
O_R	O_T	O_R	O_R	O_R
X_I	O_T	O_R	$O_R:X_I$	$O_R:X_I$
X_P	O_T	O_R	$O_R:X_I$	$X_I:X_P$

O_T = true zero X_I = imprecise non-zero

O_R = relative zero X_P = precise non-zero

Table 2

ABC Division

	O_T	O_R	X_I	X_P
O_T	∞	∞	∞	∞
O_R	O_T	O_R	O_R	O_R
X_I	O_T	O_R	$O_R:X_I$	$O_R:X_I$
X_P	O_T	O_R	$O_R:X_I$	$X_I:X_P$

O_T = true zero X_I = imprecise non-zero

O_R = relative zero X_P = precise non-zero

The addition of a relative zero and non-zero may have either an imprecise result or a relative zero. If the exponent of the non-zero operand is greater than or equal to the exponent of the relative zero, then the sum is imprecise. In particular, if the non-zero operand is precise, then the sum is the rounded imprecise representation of the precise non-zero operand; if the non-zero operand is imprecise, then the sum is the imprecise non-zero operand adjusted (signifying loss of significant digits) by the difference of exponents of the non-zero operand and relative zero. Clearly, if the exponent of the relative zero is greater than the exponent of the non-zero operand, then the sum is just the relative zero.

Table 3

ABC Addition and Subtraction

	O_T	O_R	X_I	X_P
O_T	O_T	O_R	X_I	X_P
O_R	O_R	O_R	$O_R:X_I$	$O_R:X_I$
X_I	X_I	$O_R:X_I$	$O_R:X_I$	$O_R:X_I$
X_P	X_P	$O_R:X_I$	$O_R:X_I$	$O_T:X_I:X_P$

O_T = true zero X_I = imprecise non-zero

O_R = relative zero X_P = precise non-zero

The rules just described for significant digit are arithmetic implemented in assembly language as subroutines. The entry point names are @ OL2ADD, @ OL2SUB, PIMPY, and PIDIV. (See Appendix B for the program listings.) Each routine employs a round on the digit to shift off due to unnormalization of the result to the correct number of significant digits.

There are some problems with significant digit arithmetic using unnormalized numbers. One aspect will be discussed here and others will be treated in Chapter 5. The significance rule for multiplication and division calls for retention of the same number of significant digits as in the operand with the fewest number of significant digits. This rule can result in the loss of significant digits. For instance, consider the following multiplication: 01 x 09 = 09 and 09 * 09 = 81. In both cases, the rule specifies one significant digit. For the first case, it is correct; however, in the second case the second digit "1" would be discarded. Thus, a corrective shift or an auxiliary rule is necessary for multiplication and division. The auxiliary rule for multiplication is: if $e_3 = e_1 + e_2$, then $m_3 = m_3 - 1$, where e_i's are the exponents of the normalized operands. This corrective right shift for mutliplication results in gaining one significant digit. Since division is the inverse operation of multiplication, the division auxiliary rule is: if $e_3 = e_1 - e_2$, then $m_3 = m_3 + 1$. Here the corrective left shift in division results in losing a significant digit.

Routines @ OL2MPY and @ OL2DIV are identical to PIMPY and PIDIV except for the incorporation of the above auxiliary rules.

The next chapter will examine the rules of significant digit arithmetic for elementary functions using base sixteen.

CHAPTER 3

ADJUSTMENT RULES FOR FUNCTIONS

3.1 Function Error Propagation

Let F be a function of one argument and let (e_1, f_1) be the argument value at which F is to be evaluated. Clearly, this requires the specification of two functions, g and h:

$$f_2 = g(e_1, f_1)$$

$$e_2 = h(e_1, f_1)$$

such that $r^{e_2} f_2 = F(r^{e_1}, f_1)$. This relation, however, does not imply anything concerning the number of significant digits in the result (e_2, f_2) since $(e_2 + m, r^{-m} f_2)$ also satisfies it. The adjustment criteria must be based on further considerations.

Given a number (e_1, f_1), let f_1 be regarded as an approximation to some true value f_1^T. The error in the function of the argument is defined by $\delta_1 = f_1 - f_1^T$, while the error in fraction of the function is defined by $\delta_2 = g(e_1, f_1) - g(e_1, f_1^T)$. If we use the generalized mean value theorem, then we have the estimate $|\delta_2| = \alpha|\delta_1|$. The error propagation associated with a given g is thus expressible in the terms of the "amplification factor" α. This is the procedure which is described by Ashenhurst (4).

Adjustment of the function value (e_2, f_2) is made so that α is within a factor of 16 (since we are using base sixteen) of

of any predetermined value. If α is approximately one, then an adjustment is said to be in accordance with a "significant digit" criterion. Notice that this does not imply that all the computed digits of f_2 are actually significant. Rather, it means that the error in f_2 due to the error in f_1 is of the same order of magnitude in terms of the adjustment used. Both f_1 and f_2 contain roughly the same number of "non-significant" digits.

3.2 Propagation of Relative Error: Richtmyer Adjustment Rule

Suppose x is a number approximating a true value x^T not equal to zero. The relative error between the approximate and true values may be defined as $r(x) = (x - x^T)/x^T$. Similarly, the relative error in the function F may be defined as $r_F(x) = (F(x) - (F(x^T))/F(x^T)$. If F(x) is differentiable at x, then $r_F(x) = (F'(x)\ (x - x^T))/F(x^T)$. Using the definition of r(x), we have $r_F(x) = (x^T F'(x) r(x))/F(x^T)$. If $(x - x^T)$ is small, we replace x^T with x to get:

$$r_F(x) = \frac{x\ F'(x)}{F(x)}\ r(x).$$

By definition $16^{-m-1} \leq |f| < 16^{-m}$, thus it follows that $16^m|\delta| < |\frac{\delta}{f}| < 16^{m+1}|\delta|$. Since this inequality may be applied to both the argument and function values, using respectively f_1, m_1, $\ _1$, and f_2, m_2, $\ _2$, we can write

$$16^{m_2-m_1-1}\ \left|\frac{\delta_2}{\delta_1}\right| < \left|\frac{\delta_2 f_1}{\delta_1 f_2}\right| < 16^{m_2-m_1+1}\ \left|\frac{\delta_2}{\delta_1}\right|.$$

Thus,

$$\left|\frac{\delta_2 f_1}{\delta_1 f_2}\right| = \left|\frac{16^{e_2}\delta_2}{16^{e_2} f_2} \times \frac{16^{e_1} f_1}{16^{e_1}\delta_1}\right|$$

$$= \left|\frac{F(x) - F(x^T)}{F(x)} \times \frac{x}{x - x^T}\right|$$

$$\cong \left|\frac{r_F(x)}{r(x)}\right|$$

provided that δ_1 and δ_2 are sufficiently small. An approximate inequality which holds to a first order approximation may be formed:

$$\frac{\alpha}{16} < 16^{m_1-m_2}\left|\frac{x\ F'(x)}{F(x)}\right| < 16\alpha$$

which is equivalent to

$$16^{m_1-m_2-1}\left|\frac{x\ F'(x)}{x}\right| < \alpha < 16^{m_1-m_2+1}\left|\frac{x\ F'(x)}{x}\right|$$

The above inequality bounds α as a function of the argument x and the adjustment (m_2-m_1). Taking the base 16 logarithm of all members of the inequality (this is permissible since $\log_{16}$ is a monotonic increasing function) gives

$$\log_{16}\left|\frac{x\ F'(x)}{F(x)}\right| - \Delta m - 1 < \log_{16}\alpha < \log_{16}\left|\frac{x\ F'(x)}{F(x)}\right| - \Delta m + 1.$$

With a "significant digit" criterion $\alpha=1$ the above inequality can be written as

$$-1 < \log_{16}\left|\frac{x\ F'(x)}{F(x)}\right| - \Delta m < 1.$$

Thus, an approximate adjustment criterion is

$$\Delta m \approx \log_{16} \left| \frac{x\, F'(x)}{F(x)} \right| .$$

R. D. Richtmyer had suggested the above adjustment rule for unnormalized arithmetic as well as for functions. The above deviation is identical to the one of Ashenhurst (4) except base 2 was replaced by base 16.

Notice that in actuality Δm is between -1 and +1 (base 16), a defect which Ashenhurst points out can be troublesome since it implies that $16^{-1} < \alpha \leqq 16$. He shows, however, that if it is assumed that $16^{m_1} f_1$ is uniformly distributed in the interval $(16^{-1}, 16)$ the expected value of α is unity. Therefore, on the average, Richtmyer's Rule provides the optimal adjustment.

3.3 Application of Richtmyer's Adjustment Rule

This section investigates the adjustment rules for various functions in base 16. The difference between corresponding function adjustment rules for base 16 and base 2 will be explained.

Raising to a power, $F(x) = x^n$, where n is any number gives:

$$\Delta m = \log_{16} \left| \frac{x(nx^{n-1})}{x^n} \right| = \log_{16} |n|$$

Since for the most practical purposes $|n| << 16$, an approximate adjustment rule is $\Delta m = 0$. For n=1 and n=2, the adjustment rule is $m_2 = m_1$, which means that the number of leading zeroes in the normalized fraction remains unchanged. For the square root function where n=1/2, the adjustment rule must be also $m_2 = m_1$ to conform to that of square function. However, in base 2,

$\Delta m = \log_2 |n|$ dictates $m_2=m_1-1$ for $n=1/2$ and $m_2=m_1+1$ for $n=2$. Therefore, the adjustment in base 16 is relatively coarse compared to base 2. That is, in base 2 we would actually adjust m for small values of n, whereas in base 16 the adjustment would be unnecessary.

For exponentials and logarithms, $F(x) = a^x=e^{x \ln a}$, Richtmyer's Rule specifies

$$\Delta m = \log_{16}\left|\frac{x(a^x \ln a)}{a^x}\right|$$

$$= \log_{16}|x| + \log_{16}|\ln a|$$

Recalling $16^{-m_1-1} \leq |f_1| < 16^{-m_1}$, the adjustment Δm may be expressed as either $\Delta m = (e_1-m_1-1) + \log_{16}|\ln a|$ or as $\Delta m = (e_1-m_1) + \log_{16}|\ln a|$. Once again, if a uniform distribution of $16^{m_1}f_1$ is assumed, these two equations may be averaged to give $\Delta m = e_1-m_1-1/2 + \log_{16}|\ln a|$. Thus, the adjustment rule for an arbitrary a is $m_2 = e_1 - 1/2 + \log_{16}|\ln a|$. Clearly, the right hand side of the equation must be rounded to an integer. In a similar fashion, the logarithm function $F(x) = \log_a (x)$ where $x>0$ should be adjusted by the rule: $m_2 = e_1 + 1/2 - \log_{16}|\ln a|$. Notice that the only difference of adjustment rules for exponentials and logarithms is the factor of one-half.

More complicated functions such as trigonometric functions, hyperbolic functions, and the error function may be expressed as a power series in x. As examples, consider the following

expansions with $0 \leq x \leq \pi/4$:

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} + \ldots$$

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} + \ldots$$

If we assume as Ashenhurst does that the error in the function can be approximated by the error in the first term of the series, the sine can be thought of as a power series in x and the cosine can be thought of as a power series in x^2. Using the previous derived rule for $F(x) = x^n$, namely,

$$m_2 = m_1 + \log_{16} |n|$$

we have for sine (n=1) $m_2=m_1$ and for cosine (n=2) $m_2 = m_1 + \log_{16} 2$. The last adjustment rule can be approximated by $m_2=m_1$. Ashenhurst derived the cosine adjustment in base 2 as $m_2 = 2e_1 - m_1$. Again, this difference is due to the different bases used. Using the first term for an approximation, the adjustment rule $m_2=m_1$ also holds for hyperbolic functions, exponential functions, and the error functions.

Significant digit function routines have been implemented as subroutines which call the corresponding IBM PL/1 library functions. Ashenhurst in (4) suggested this strategy of carrying out the function computation by normalized and then adjusting the result at the end. The adjustment rules for imprecise arguments for functions in base 16 falls into two equations--either $m_2=m_1$ or $m_2= |e_1|$.

Table 4

ABC Function Rules

Function	Base 2 Adjustment	Base 16 Adjustment
Sin	$m_2=m_1$	$m_2=m_1$
Cos	$m_2=\|2e_1-m_1\|$	$m_2=m_1$
Tan	$m_2=m_1$	$m_2=m_1$
Square Root	$m_2=m_1-1$	$m_2=m_1$
Exponent: e^x	$m_2=\|e_1-1\|$	$m_2=m_1$
Log_e	$m_2=\|e_1\|$	$m_2=\|e_1\|$
Log_{10}	$m_2=\|e_1+1\|$	$m_2=\|e_1\|$
Log_2	$m_2 \ \|e_1\|$	$m_2=\|e_1\|$
Sinh	$m_2=m_1$	$m_2=m_1$
Cosh	$m_2=\|2e_1-m_1\|$	$m_2=m_1$
Tanh	$m_2=m_1$	$m_2=m_1$
Arctan	$m_2=m_1$	$m_2=m_1$
Error Function	$m_2=m_1$	$m_2=m_1$
Complement Error Function	$m_2=m_1$	$m_2=m_1$

The previous discussion assumed the arguments were imprecise or true zero. The subroutines (except for the square root) always return an imprecise number even when the argument is precise. The justification for this action is that values of the function can only rarely be represented as a precise,

number, for instance, sin 30° = .5 and cos 0° = 1. For a relative zero argument, the argument exponent is examined. If the exponent is positive, the same relative zero is returned for the function value. Relative zeroes with non-positive exponents are treated as if the argument is near zero. In particular, an imprecise one with one significant digit is returned for the exponential, cos, cosh, and error functions, while a relative zero with exponent of +14 is returned for the sine, tan, sinh, tanh, arctan, and complement error functions. An error condition is signaled when relative zeroes with non-positive exponents are arguments to the log functions. For the square root function, the same relative zero is returned as the functional value. This implementation of relative zero as an argument for the above functions does not account for repeated functional evaluations with relative zero. Thus, the treatment of relative zero should be studied further and refined.

This completes the discussion of implemented significant digit adjustment rules for functions. The next chapter is concerned with the input and output convention for precise and imprecise numbers.

CHAPTER 4

INPUT AND OUTPUT

This chapter will deal with the problems of input and output. We shall also discuss algorithms for number base conversion as described by Kanner (13) for significant digit arithmetic.

4.1 Objectives of Significant Digit I/O

The main objective of I/O for significant digit arithmetic is to obtain a faithful representation of significance under the operation of base conversion from binary to decimal, as well as the inverse of this operation. For instance, the number 2.00 on input should be immediately converted on output (without any intervening computation) as 2.00, and not, say, as 1.99. Kanner points out this can be achieved only if at least one guard bit is maintained in the floating point fraction. In this implementation, as already pointed out, a guard byte (4 bits) is always available. This multipurpose guard byte, whose bits are considered to be nonsignificant, also serves to separate accumulated rounding error from the error inherent to the computation. Since I/O is part of the human-computer interfacing problem, there should be ample diagnostic aids to help the user interpret the computer outputs. The I/O routines of the present work were written with the above objectives in mind.

4.2 Decimal to Binary Conversion

The following algorithms of Kanner (13) are based upon the strategy that correct significance can be retained in the conversion of integers. Briefly, the procedure involves multiplying the number by that power of ten which is necessary to express the significant portion as an integer, then this integer is converted to its intermediate internal representation in the desired number system, and finally, this result is divided by the same power of ten.

The following discussions have been adapted to base 16 instead of base 2 as discussed in Kanner (13). The binary representation of N can be obtained from the decimal integer representation $w_1 w_2 \ldots w_{n-1} w_n$ by considering the following scheme: $a_o=0$, $a_i=10a_{i-1} + w_i$, $N=a_n$. The long floating point number (64 bits) implementation dictates, for efficiency, the use of floating point hardware instead of the general registers (32 bits). Each w_i can be represented as a floating point number with leading zeroes up to the right most hexadecimal digit and with an exponent of C, where C is the total number of fractional digits. (For the 360/370 long floating point number, C is 14.) This scheme, in effect, allows fixed point (integer) arithmetric to be performed with floating point hardware.

We now consider the algorithm to convert any decimal number to binary while retaining significant digits. Let $N = \pm x_1 x_2 \ldots x_j . y_1 y_2 \ldots y_k * 10^E$. Clearly, (j+k) must be less or equal to D, the maximum number of decimal digits permitted.

(D is 15 for 370/360). The signs of N and E are saved. Next, the integers $|E|$, k, and $x_1x_2\cdots x_jy_1y_2\cdots y_R$ are converted to binary using the previous algorithm. If it is possible to introduce a guard digit, then I, the binary representation of $x_1x_2\cdots x_jy_1y_2\cdots y_R$ is then adjusted.

The next step is to multiply I by the quantity 10^{E-k} using significant digit (SD) arithmetic. In this way, the final number will have the same significance as the original number. This step can be separated into three cases, depending upon the sign and magnitude of E. The cases are:

(1) $E \geq 0$. Let ES = I-k. For $ES < 0$, divide I by $10^{|ES|}$. For $0 \leq ES < M$, where M is the maximum representable power of ten, multiply I by $10^{|ES|}$. (Because of unnormalization, M is 49 for an imprecise number while M is 63 for a precise number.)

(2) $-M \leq E < 0$. Divide I by $10^{|E|}$ and then divide the result by 10^k.

(3) $E < -M$. Divide I successively by 10^M, $10^{|E|-M}$, and 10^k. In this case, E has the lower bound of smallest representable exponent. (For the present work, E must be equal or greater than -63.)

All the operations in the above three cases are performed in significant digit arithmetic. The last step is to attach the proper algebraic sign to the number.

4.3 Binary to Decimal Conversion

Let g be the number of guard digits in the fractional part of the floating point number. (In the present work, imprecise numbers have one guard digit and precise numbers have none.)

The first step is to convert the original number to a significant integer multiplied by a power of ten. There are three cases depending on x, the exponent of the number to be converted.

(1) $x \geq (C-g)$. Let $ES = x - (C-g)$ and E(n) denote the exponent of the normalized binary representation of 10^n. The original number is divided by 10^n, where n is chosen such that $E(n) - 1 \leq ES < E(n+1)$. If the exponent of the quotient is less than (C-g), then the number is divided instead by 10^{n-1}. After conversion of the significant integer to decimal, the result is the decimal integer times 10^n or 10^{n-1}.

(2) $(C-g) - G \leq x < (C-g)$, where G is the maximum representable exponent in the floating point number. The original number is multiplied by 10^n, where n is chosen such that $E(n-1) < ES \leq E(n)$. If the exponent of the product is less than (C-g), the number is mutliplied instead by 10^{n+1}. The result is the converted integer times 10^{-n} or $n0^{-n-1}$.

(3) $x < (C-g) - G$. The number is multiplied by 10^M, where M is the maximum representable power of ten. Then the resulting product is treated as in case (2) above. Here the result is the converted integer times 10^{-p-M}, p is the power of ten used in the second multiplication.

Finally, if a decimal point is to be inserted in any desired position, the power of ten is modified accordingly.

4.4 The I/O Routines

The input and output of imprecise numbers conform to the conventions that are used for PL/1 E and F formats. The special symbol P is used to distinguish precise numbers from imprecise numbers, e.g., 1.00P and -1.23P+01. Like PL/1, the maximum number of digits per item allowed is 15, due to the computer word length. The I/O routines consist of three modules: OL2PILK, which accepts the input numbers as character strings in the PL/1 GET LIST manner, OL2PICF, which converts character strings into long floating number, and OL2PIFD, which converts the internal numbers into character strings for output. Both OL2PICF and OL2PIFD perform an operation with a ten of power, as described by Kanner, in two successive steps in the ten's place and then in the one's place to preserve precise numbers. It was found that a precise number multiplied by 10^{23} became imprecise, but the same precise number multiplied by 10^{20} and 10^{3} remained precise. This is a result of the computer word length used.

Besides returning the correct significance for a number on output, OL2PIFD also supplies other information to interpret the number. There is significance pointer to indicate the last significant digit of the output number. This is useful, for example, when a user does not want to output all significant digits of a number. A common user error is to specify an insufficient field width for output. In this case an error flag is returned along with as much of the number as possible. For

output of arrays it is also useful to have indicators for a maximum significant digit count and a maximum field width. This allows the user to analyze his array output with respect to maximum significance and field width. Each of these considerations are valuable and contribute to a reliable and useful significant digit I/O package.

CHAPTER 5
DISCUSSION

Precise and imprecise arithmetic, as described up to this point, can be used to carry out more complicated mathematical operations. However, separate routines to handle operations with complex numbers and matrices may be advisable. Further investigation is needed to study the effects of correlated errors with imprecise numbers. More significance rules may be required or perhaps, the present significant rules must be modified.

5.1 Matrix

Most matrix operations consist of a series of inner products of vectors. The inner product subroutine OL2PIVP was written to extend ABC arithmetic to matrix operations. Following the same approach of the significant functions, the addition and multiplication are done normalized. At the end, the result is unnormalized to the correct number of significant digits by keeping a count of the maximum number of leading zeroes encountered. In this way, the resulting inner product is more accurate because of less round-off errors with normalized numbers.

5.2 Complex Numbers

A complex number can be represented as a number with real and imaginary parts. Routines using the approach of the inner product can extent ABC arithmetic to include complex numbers. Some definitions are necessary: a complex number is precise if

and only if both real and imaginary parts are precise; the significance of a complex number is that of the less significant part; a complex true zero consists of two real true zeroes. The possibility of a relative zero part and non-zero part must be studied and appropriate definitions implemented.

5.3 Some Problems with Significant Digit Arithmetic

The adjustment rules for imprecise numbers are based on the assumption that their associated errors are statistically independent. If their errors are correlated, the resulting imprecise number may have a representation which is not a reliable measure of significance. An example of correlated error is found in the multiplication of an imprecise number by itself. The resulting product should be more significant since the approximation error was squared. The auxiliary rule for multiplication attempts to correct this situation by increasing the number of significant digits by one. Similarly, repeated summation of an imprecise number by itself, under the present adjustment rule, would allow the number of significant digits to increase. A combination of these correlated errors can be found in the vector norm operation. Thus, correlated errors in significant digit arithmetic should be studied.

To complete the implementation of significant digit arithmetic, routines to compare two operands are needed. Detection of a relative and true zero is done by check for zero fraction. If the exponent is also all zero, then the number is a true zero. To compare two relative zeroes, the exponents are examined.

Since relative zeroes have no significant digits, only the exponents of a non-zero and a relative zero can be compared. True zero would always be "lower" than any number which is not a true zero. Care must be taken to avoid using the guard bits of an imprecise number in a compare. After removing the guard bits and normalizing, comparison of imprecise numbers can be made. Finally, precise numbers are compared with any special treatment.

As noted in Chapter 3, the present implementation of functional values for relative zero arguments did not account for the possibility of repeated function evaluations using relative zeroes. Each function should be studied separately to provide a more comprehensive treatment of relative zero arguments and correlated errors in functional evaluations in significant digit arithmetic.

There are other methods of error monitoring. The interval arithmetic of Moore (19) also provides error bounds on computed results. Metropolis (17) gives techniques of analyzing algorithms in unnormalized arithmetic. Computational errors are also discussed in (21). Metropolis (9) states ". . . different algorithms for the same problem give different representations is connected with the observation that mathematically equivalent approaches are not computationally equivalent." Significant digit arithmetic is one way to aid the error analysis of a computational algorithm. It is hoped that present implementation will aid further study and experimentation with imprecise numbers and significant digit arithmetic.

REFERENCES

(1) Ashenhurst, R. L., and N. Metropolis. "Error Estimation in Computer Calculation." The Am. Math. Monthly, Vol. 72, No. 2, Part II (1959), 47-58.

(2) Ashenhurst, R. L., and N. Metropolis. "Unnormalized Floating Point Arithmetic." J. ACM, 6 (1959), 415-428.

(3) Ashenhurst, R. L. "The MANIAC III Arithmetic System." Proc. Joint Computer Conf., 21 (1962), 195-202.

(4) Ashenhurst, R. L. "Function Evaluation in Unnormalized Arithmetic." J. ACM, 11 (1964), 168-187.

(5) Ashenhurst, R. L. "Number Representation and Significance Monitoring." Mathematical Software. Edited by J. R. Rice. New York: Academic Press, 1971, 67-86.

(6) Blandford, R. C., and N. Metropolis. "The Simulation of Two Arithmetic Structures." LA-3979, Los Alamos Scientific Laboratory (1968).

(7) Bright, H. S., B. A. Colhoun, and F. B. Mallory. "A Software System for Tracing Numerical Significance during Computer Program Execution." SJCC, (1971), 387-392.

(8) Fraser, M., and N. Metropolis. "Algorithms in Unnormalized Arithmetic III. Matrix Inversion." Num. Math., 12 (1968), 416-428.

(9) Gardiner, V., and N. Metropolis. "A Comprehensive Approach to Computer Arithmetic." LA-4531, Los Alamos Scientific Laboratory (1970).

(10) Gardiner, V., and N. Metropolis. "Significant Digit Arithmetic on a CDC 6600." LA-4470, Los Alamos Scientific Laboratory (1970).

(11) Goldstein, M., and Hoffberg, S. "The Estimation of Significance." Mathematical Software. Edited by J. R. Rice. New York: Academic Press, 1971, 93-104.

(12) Gray, H. L., and C. Harrison, Jr. "Normalized Floating-Point Arithmetic with an Index of Significance." PJCC Vol. 16 (1959), 244-248.

(13) Kanner, H. "Number Base Conversion in a Significant Digit Arithmetic." J. ACM, 12 (1965), 242-246.

(14) Menzel, M., and N. Metropolis. "Algorithms in Unnormalized Arithmetic, II. Unrestricted Polynomial Evaluation." Num. Math., 10 (1967), 451-462.

(15) Metropolis, N., and R. L. Ashenhurst. "Significant Digit Computer Arithmetic." IRE Trans. Electron. Computers, EC-7,265 (1958).

(16) Metropolis, N. "Algorithms in Unnormalized Arithmetic, I. Recurrence Relations." Num. Math., 7 (1965), 104-112.

(17) Metropolis, N. "Algorithms in Unnormalized Arithmetic." Proc. Colloq. Inst. Centre. Nat. Rech. Sci., Besaucon, France (1967).

(18) Miller, R. H. "An Example in Significant-Digit Arithmetic." Com. ACM, Vol. 7 (1964), 21.

(19) Moore, R. E. Interval Analysis. Englewood Cliffs, N. J.: Prentice Hall, 1966.

(20) Phillips, J. R. "The Structure and Design Philosophy of OL/2 - An Array Language." Part I. (To appear.)

(21) Rall, L. B. Errors in Digital Computation. New York: John Wiley, 1965.

(22) Richtmyer, R. D. "The Estimation of Significance." AEC Research and Development Report, NYO-9083 (1960).

(23) Yasui, Toshio. "Significant Digit Arithmetic on ILLIAC IV." ILLIAC IV Report 211 (1969).

APPENDIX A

EXAMPLES OF SIGNIFICANT DIGIT ARITHMETIC

APPENDIX A

This appendix contains several examples of significant digit arithmetic. Each program illustrates how a software system may generate the necessary linkage or CALLS to the basic significant digit routines. The unnormalized significance output was provided by OL2PIFD. The significance pointer is the position of the last significant digit, counting from the leftmost position of the field. This significance pointer can also return negative numbers: -3 for true and relative zeroes, -2 for insufficient field width (severe error), and -1 to indicate that not all of the significant digits in the fraction were displayed for the given field width (warning error). The information in the significance pointer may be incorporated in a more elaborate I/O routine which can place a marker such as the underscore character under the last significant digit of an output item.

```
 1                NEWTON:  PROC OPTIONS(MAIN);
 2                     DCL OL2PILK ENTRY RETURNS ( CHAR(32) VAR );
 3                     DCL OL2PICF ENTRY (CHAR(32) VAR) RETURNS ( FLOAT BIN(53));
 4                     DCL OL2PIFD ENTRY ( FLOAT BIN (53),     /* UNNORMALIZED #   */
                                          FIXED BIN (15),     /*  FIELD WIDTH     */
                                          FIXED BIN (15),     /* FRACTIONAL DIGITS*/
                                          FIXED BIN (15),     /* EORFLAG          */
                                          FIXED BIN (15)  )   /*  UB_NUMBER       */
                                RETURNS ( CHAR(120) VAR ) ; /* NUMBER IN CHAR   */
 5                     DCL (W,D,EORFLAG,UBNUM) FIXED BIN(15), NUMBER CHAR(32) VAR;
 6                     DCL   @OL2MPY   ENTRY(FLOAT BIN(53),FLOAT BIN(53) )
                              RETURNS (FLOAT BIN(53));
 7                     DCL   @OL2DIV   ENTRY(FLOAT BIN(53),FLOAT BIN(53) )
                              RETURNS (FLOAT BIN(53));
 8                     DCL   @OL2ADD   ENTRY(FLOAT BIN(53),FLOAT BIN(53) )
                              RETURNS (FLOAT BIN(53));
 9                     DCL   (HALF,A,Z,X) FLOAT BIN(53);
10                     DCL A_CHAR CHAR(32) VAR;
11                     DCL INTHEX ENTRY (FLOAT BIN(53)) RETURNS (CHAR(16));
12                     DCL A_IN_HEX CHAR(16);
13                     DCL IHCHAR CHAR(16);
14                ON ENDFILE(SYSIN) GOTO FINIS;
16                     GET DATA (W,D,EORFLAG);
17                     PUT DATA (W,D,EORFLAG);
18                     HALF = 0.5E+00;              /* PRECISE ONE-HALF */
19                LOOP: PUT PAGE LIST('NEWTON METHOD FOR SQUARE ROOT OF A NUMBER');
20                     GET LIST(NSTEP);             /* READ IN THE NUMBER OF STEPS */
21                     A_CHAR= OL2PILK;     A = OL2PICF(A_CHAR); /* READ A P/I # */
23                     PUT SKIP(2);  PUT DATA(A_CHAR);
25                     A_IN_HEX = INTHEX(A);
26                     PUT SKIP(2);     PUT DATA(A_IN_HEX);   PUT SKIP(4);
29                     PUT EDIT ('NORMALIZED','INTERNAL HEX',
                                 'UNNORMALIZED','SIGNIFICANCE')
                              (COL(09),A,COL(40),A,COL(68),A,COL(100),A);
30                     PUT EDIT ('REPRESENTATION','REPRESENTATION',
                                 'SIGNIFICANCE REPRESENTATION','POINTER')
                              (COL(07),A,COL(39),A,COL(61),A,COL(103),A);
                  /* NEWTON'S METHOD: X = 0.5*(A/X + X); FOR X = SQRT(A); */
31                     X = A;
32                          DO I_STEP = 1 BY 1 TO NSTEP;
33                          Z = @OL2DIV(A,X);
34                          Z = @OL2ADD(Z,X);
35                          X = @OL2MPY(HALF,Z);
36                     NUMBER = OL2PIFD (  X ,   W,D,EORFLAG,UBNUM);
37                     IHCHAR = INTHEX( X );
38                     PUT SKIP(2);     PUT DATA (X);
40                     PUT SKIP(0) EDIT (IHCHAR,NUMBER,UBNUM)
                                        (COL(38),A(16),X(10),A(24),X(10),F(10) );
41                          END;
42                     GO TO LOOP;
43                     FINIS: PUT SKIP LIST ('END_OF_FILE');
44                END NEWTON;
```

NEWTON METHOD FOR SQUARE ROOT OF A NUMBER

A_CHAR='3.1415';

A_IN_HEX='4A00000000032439';

NORMALIZED REPRESENTATION	INTERNAL HEX REPRESENTATION	UNNORMALIZED SIGNIFICANCE REPRESENTATION	SIGNIFICANCE POINTER
X= 2.070747375488281E+00;	490J0000002121C8	2.07075	22
X= 1.793914794921875E+00;	490J0000001C83E0	1.79392	22
X= 1.772552490234375E+00;	490J0000001C5C60	1.77255	22
X= 1.772422790527343E+00;	490J0000001C58D8	1.77242	22
X= 1.772422790527343E+00;	490J0000001C58D8	1.77242	22
X= 1.772422790527343E+00;	490J0000001C58D8	1.77242	22
X= 1.772422790527343E+00;	490J0000001C58D8	1.77242	22
X= 1.772422790527343E+00;	490J0000001C58D8	1.77242	22
X= 1.772422790527343E+00;	490J0000001C58D8	1.77242	22
X= 1.772422790527343E+00;	490J0000001C58D8	1.77242	22

NEWTON METHOD FOR SQUARE ROOT OF A NUMBER

A_CHAR='3.141500000';

A_IN_HEX='4600000324395810';

NORMALIZED REPRESENTATION	INTERNAL HEX REPRESENTATION	UNNORMALIZED SIGNIFICANCE REPRESENTATION	SIGNIFICANCE POINTER
X= 2.07074999955296E+00;	450J002121CAC080	2.0707500000	22
X= 1.79391659210290E+00;	450J001C83E1E230	1.7939165912	22
X= 1.77255641855299 4E+00;	450J001C5C641E80	1.7725564186	22
X= 1.77242771827150 1E+00;	450J001C58DD2AC8	1.7724277183	22
X= 1.77242771349847 3E+00;	450J001C58DD2980	1.7724277135	22
X= 1.77242771361488 8E+00;	450J001C58DD2988	1.7724277136	22
X= 1.77242771349847 3E+00;	450J001C58DD2980	1.7724277135	22
X= 1.77242771361488 8E+00;	450J001C58DD2988	1.7724277136	22
X= 1.77242771349847 3E+00;	450J001C58DD2980	1.7724277135	22
X= 1.77242771361488 8E+00;	450J001C58DD2988	1.7724277136	22

NEWTON METHOD FOR SQUARE ROOT OF A NUMBER

A_CHAR='3.141592653';

A_IN_HEX='46000003243F6A86';

NORMALIZED REPRESENTATION	INTERNAL HEX REPRESENTATION	UNNORMALIZED SIGNIFICANCE REPRESENTATION	SIGNIFICANCE POINTER
X= 2.07079632650 1294E+00;	450J0021 21F85430	2.0707963265	22
X= 1.79394515615 4222E+00;	4500001C 83FFC608	1.7939451562	22
X= 1.77258258266 3744E+00;	450J001C 5C7F8DE0	1.7725825827	22
X= 1.77245385537 3896E+00;	450J001C 58F892E8	1.7724538554	22
X= 1.77245385071 7283E+00;	450J001C 58F891A8	1.7724538507	22
X= 1.77245385071 7283E+00;	450J001C 58F891A8	1.7724538507	22
X= 1.77245385071 7283E+00;	450J001C 58F891A8	1.7724538507	22
X= 1.77245385071 7283E+00;	450J001C 58F891A8	1.7724538507	22
X= 1.77245385071 7283E+00;	450J001C 58F891A8	1.7724538507	22
X= 1.77245385071 7283E+00;	450J001C 58F891A8	1.7724538507	22

```
1                SINHPGM:   PROC OPTIONS(MAIN);
2                     DCL OL2PILK ENTRY RETURNS ( CHAR(32) VAR );
3                     DCL OL2PICF ENTRY (CHAR(32) VAR) RETURNS ( FLOAT BIN(53));
4                     DCL DL2P)FD ENTRY ( FLOAT BIN (53),     /* UNNORMALIZED #   */
                                         FIXED BIN (15),     /*  FIELD W)DTH     */
                                         FIXED BIN (15),     /* FRACTIDNAL DIGITS*/
                                         F)XED BIN (15),     /* EDRFLAG          */
                                         FIXED B)N (15)  )   /*  UB_NUMBER       */
                                RETURNS ( CHAR(120) VAR ) ; /* NUMBER IN CHAR   */
5                     DCL (W,D,EORFLAG,UBNUM) F(XED BIN(15), NUMBER CHAR(120) VAR;
6                     DCL   @OL2ADD   ENTRY(FLOAT BIN(53),FLOAT BIN(53) )
                                      RETURNS (FLOAT B)N(53));
7                     DCL   @OL2SUB   ENTRY(FLOAT B)N(53),FLOAT BIN(53) )
                                      RETURNS (FLDAT B)N(53));
8                     DCL   @DL2MPY   ENTRY(FLOAT B(N(53),FLDAT BIN(53) )
                                      RETURNS (FLOAT B(N(53));
9                     DCL  T(11) FLDAT BIN(53) STAT(C;   /* TEMP VARIABLES */
10                    DCL   (X,Y,PTWD,Z)        FLOAT B)N(53) STATIC;
11                    DCL X_CHAR CHAR(32) VAR, Y_CHAR CHAR(32) VAR;
12                    DCL @LSHLS ENTRY ( FLOAT B)N(53) ) RETURNS ( FLOAT BIN(53) );
13                    DCL @LSHLC ENTRY ( FLDAT B)N(53) ) RETURNS ( FLOAT BIN(53) );
14                    DCL )NTHEX ENTRY (FLDAT BIN(53)) RETURNS (CHAR(16));
15                    DCL IHCHAR CHAR(16);
16                    DCL X_INTHEX CHAR(16), Y_INTHEX CHAR(16);
17               ON ENDF)LE(SYS)N) GDTD FIN)S;
19                    GET DATA (W,D,EDRFLAG);
20                    PUT DATA (W,D,EORFLAG);
21                    PTWO = 2.0E+00;      /* PREC)SE TWD */
22               LOOP: PUT PAGE L)ST('SINH(X+Y)*SINH(X-Y) = 2.0P*COSH(Y)*SINH(X)');
23                    X_CHAR = OL2P)LK;    X = OL2PICF(X_CHAR);
25                    Y_CHAR = OL2P)LK;    Y = OL2PICF(Y_CHAR);
27                    X_(NTHEX = )NTHEX(X);
28                    Y_)NTHEX = )NTHEX(Y);
29                    PUT SKIP(2);PUT DATA(X_CHAR,X_INTHEX);
31                    PUT SKIP(2);PUT DATA(Y_CHAR,Y_INTHEX);
                      /* SINH ID:  S)NH(X+Y) *SINH(X-Y) = 2.0P*COSH(Y)*SINH(X); */
33                    T(1) = @OL2ADD(X,Y);
34                    T(2) = @LSHLS( T(1) );     /* SINH */
35                    T(3) = @OL2SUB(X,Y);
36                    T(4) = @LSHLS( T(3) );     /* SINH */
37                    T(5) = @LSHLC(Y);          /* COSH */
38                    T(6) = @LSHLS(X);          /* SINH */
39                    T(7) = @OL2MPY(PTWO,T(5));
40                    T(8) =  @OL2MPY(T(7),T(6));   /*  RHS  */
41                    T(9) = @DL2ADD( T(2),T(4) );  /* LHS */
42                    T(10) = @OL2SUB (T(8),T(9) );
43                    PUT SK)P(2);
44                    PUT ED)T ('NORMALIZED','INTERNAL HEX',
                                'UNNORMALIZED','SIGNIFICANCE')
                              (COL(09),A,COL(40),A,COL(68),A,COL(100),A);
45                    PUT EDIT ('REPRESENTATION','REPRESENTATION',
                                'SIGNIFICANCE REPRESENTATION','POINTER')
```

```
                        (COL(07),A,COL(39),A,COL(61),A,COL(103),A);
   46                   DO I = 1 BY 1 TO 10;
   47                   NUMBER = OL2P(FD ( T(I),  W,D,EORFLAG,UBNUM);
   48                   PUT SKIP(2);    PUT DATA (T(I));
   50                   (HCHAR = INTHEX( T(I));
   51                   PUT SKIP(0) EDIT ((HCHAR,NUMBER,UBNUM)
                                 (COL(38),A(16),X(10),A(24),X(10),F(10) );
   52                   END;
   53                   GOTO LOOP;
   54                   FINIS: PUT SKIP LIST ('END_OF_FILE');
   55               END SINHPGM;
```

SINH(X+Y)+SINH(X-Y) = 2.0P*COSH(Y)*SINH(X)

X_CHAR='-2.71' X_INTHEX='CB0000J000002B5C';

Y_CHAR='-3.14' Y_INTHEX='CB000000000003230';

NORMALIZED REPRESENTATION	INTERNAL HEX REPRESENTATION	UNNORMALIZED SIGNIFICANCE REPRESENTATION		SIGNIFICANCE POINTER
T(1)=-5.849853515625000E+00;	CB0J000000005099	-5.850	E+00	8
T(2)=-1.735898437500000E+02;	CC0000000000AD97	-1.7359	E+02	9
T(3)= 4.299316406250000E-01;	4B0J000000000E1	4.30	E-01	7
T(4)= 4.433593750000000E-01;	4B0J000000000718	4.43	E-01	7
T(5)= 1.157226562500000E+01;	4B0J000000008928	1.1572	E+01	9
T(6)=-7.480957031250000E+00;	CB0J000000007782	-7.481	E+00	8
T(7)= 2.314453125000000E+01;	4B0J000000017250	2.3145	E+01	9
T(8)=-1.731445312500000E+02;	CC0J000000000A025	-1.7314	E+02	9
T(9)=-1.731445312500000E+02;	CC0J000000000AD25	-1.7314	E+02	9
T(10)= 0.000000000000000E+00;	4C00000000000000	0.000000000000E+14		-3

SINH(X+Y)+SINH(X-Y) = 2.0P*COSH(Y)*SINH(X)

X_CHAR='100.99' X_INTHEX='4800000000064FD7';

Y_CHAR='-.45' Y_INTHEX='CB0000000000733';

	NORMALIZED REPRESENTATION	INTERNAL HEX REPRESENTATION	UNNORMALIZED SIGNIFICANCE REPRESENTATION		SIGNIFICANCE POINTER
T(1)=	1.00540039062500E+02;	4800000000064BA4	1.00540	E+02	10
T(2)=	2.306501939463504E+43;	6E0000000001C8C6	2.3065	E+43	9
T(3)=	1.014399414062500E+02;	4B00000000065704	1.01440	E+02	10
T(4)=	5.672541084808736E+43;	6E0000000002BB2D	5.673	E+43	8
T(5)=	1.101562500000000E+00;	4C000000000011A	1.10	E+00	7
T(6)=	3.617133503E96191E+43;	6E000000000019F3A	3.6171	E+43	9
T(7)=	2.203125000000000E+00;	4C000000000000234	2.20	E+00	7
T(8)=	7.970774162756061E+43;	7000000000000393	8.0	E+43	6
T(9)=	7.979043024272240E+43;	6E000C000000393F3	7.979	E+43	8
T(10)=	0.000000000000000E+00;	7000000000000000	0.000000000000E+57		-3

SINH(X+Y)+SINH(X-Y) = 2.0P*COSH(Y)*SINH(X)

X_CHAR='.1P' X_INTHEX='4101999999999999A';

Y_CHAR='.2P' Y_INTHEX='4103333333333333';

NORMALIZED REPRESENTATION	INTERNAL HEX REPRESENTATION	UNNORMALIZED SIGNIFICANCE REPRESENTATION	SIGNIFICANCE POINTER
T(1)= 3.0C0000C000000000E-01;	4104CCCCCCCCCCCD	3.000000000000000E-01	19
T(2)= 3.045202934471420E-01;	41040F50ABD52E4A	3.04520293447143E-01	19
T(3)=-9.999999999999985E-02;	C1019999999999999	-1.000000000000000E-01	19
T(4)=-1.001667500198439E-01;	C1019A4873378598	-1.001667500198441E-01	19
T(5)= 1.020066755619076E+00;	4201052318481EEA	1.0200667556190BE+00	19
T(6)= 1.001667500198441E-01;	41019A4873378598	1.001667500198441E-01	19
T(7)= 2.040133511238153E+00;	42020A4630963DD4	2.040133511238153E+00	19
T(8)= 2.043535434272991E-01;	4103450838907881	2.043535434272991E-01	19
T(9)= 2.043535434272987E-01;	4103450838907BAF	2.043535434272991E-01	19
T(10)= 4.440892098500626E-16;	4100000000000002	1. E-15	4

```
1                MATR)XMPY: PROC OPTIONS(MAIN);
2                DCL OL2PIVP ENTRY I FLOAT BIN(53),     /*   XI1)                */
                                    FLOAT BIN(53),     /*   YI1)                */
                                    FIXED B)N(15) )    /*   LENGTH OF VECTORS   */
                              RETURNS I FLOAT BIN(53) ); /*   INNER PRODUCT       */
3                     DCL OL2P)LK ENTRY RETURNS ( CHARI32) VAR );
4                     DCL OL2P)CF ENTRY (CHAR(32) VAR) RETURNS ( FLOAT BINI53));
5                     DCL INPUT_CHAR CHAR(32) VAR;
6                     DCL OL2PIFD ENTRY ( FLOAT BIN I53),     /* UNNORMALIZED #   */
                                          FIXED BIN (15),     /*  F)ELD WIDTH     */
                                          FIXED B)N (15),     /* FRACTIONAL DIGITS*/
                                          FIXED B)N I15),     /* EORFLAG          */
                                          F)XED B)N (15)  )   /*  UB_NUMBER       */
                                    RETURNS ( CHAR(120) VAR ) ; /* NUMBER IN CHAR   */
7                     DCL (W,D,EORFLAG,UBNUM) F)XED BIN(15), NUMBER CHAR(120) VAR;
8                     DCL INTHEX ENTRY (FLOAT BIN(53)) RETURNS ICHAR(16));
9                     DCL IHCHAR CHAR(16);
10                    DCL INPUT_)N_HEX CHAR(16);
11               ON ENDFILEISYSIN) GOTO FINIS;
13                    GET DATA (W,D,EORFLAG);   PUT DATA (W,D,EORFLAG);
15               LOOP:  PUT PAGE L)ST('MATR)X MULTIPLY US)NG THE OL2PIVP ROUTINE');
16                    GET LIST(N);  /* THE ORDER OF THE SQUARE MATRICES */
17                    NSQ = N*N;
18                    PUT SK)P(2);   PUT DATA(N);
                 /* MATR)X_A * MATRIX_B = MATRIX_C; */
20                    DCL A(256)   FLOAT B)N(53);
21                    DCL B(256)   FLOAT BIN(53);
22                    DCL C(256)   FLOAT BIN(53);
                      /* NOTE: READ IN MATR)X A ROW-W)SE THEN B COLUMN-WISE. */
23                    PUT SK)PI3) L)ST I'THE ELEMENTS OF MATRIX A ARE:');
24                    DO I = 1 BY 1 TO NSQ;
25                    INPUT_CHAR = OL2PILK;     AII) = OL2PICF(INPUT_CHAR);
27                    )NPUT_)N_HEX = INTHEX(A(I));
28                    PUT SKIPI2);   PUT DATA(INPUT_CHAR,INPUT_IN_HEX);
30                    END;
31                    PUT SKIPI3) LIST I'THE ELEMENTS OF MATRIX B ARE:');
32                    DO ) = 1 BY 1 TO NSQ;
33                    )NPUT_CHAR = OL2PILK;     B(I) = OL2P)CFI)NPUT_CHAR);
35                    )NPUT_)N_HEX = INTHEX(BII));
36                    PUT SKIP(2);   PUT DATA(INPUT_CHAR,INPUT_IN_HEX);
38                    END;
39                    PUT PAGE LIST('MATR)X MULTIPLY USING THE OL2PIVP ROUTINE');
40                    ISTEP =1 ;
41                    DO I = 1 BY N TO NSQ;
42                          DO J = 1 BY N TO NSQ;
43                          CIISTEP) = OL2PIVP I A()), B(J), N ) ;
44                          ISTEP = ISTEP +1 ;
45                          END;
46                    END;
47                    PUT SK)P(4);
48                    PUT ED)T ('NORMAL)ZED',')NTERNAL HEX',
                                'UNNORMALIZED','SIGNIFICANCE')
```

```
                    (COL(09),A,COL(40),A,COL(68),A,COL(100),A);
49                  PUT EDIT ('REPRESENTATION','REPRESENTATION',
                              'SIGNIFICANCE REPRESENTATION','POINTER')
                    (COL(07),A,COL(39),A,COL(61),A,COL(103),A);
50                  DO I = 1 BY 1 TO NSQ;
51                  NUMBER = OL2P(FD(C(I),W,D,EORFLAG,UBNUM);
52                  IHCHAR = INTHEX(C(I));
53                  PUT SK(P(2);    PUT DATA (C(I));
55                  PUT SKIP(0) EDIT ((IHCHAR,NUMBER,UBNUM)
                                (COL(38),A(16),X(10),A(24),X(10),F(10) );
56                  END;
57                GOTO LOOP;
58                  FINIS: PUT SKIP LIST ('END_OF_FILE');
59                END MATRIXMPY;
```

MATRIX MULTIPLY USING THE DL2PIVP ROUTINE

N= 2;

THE ELEMENTS OF MATRIX A ARE:

INPUT_CHAR='.1P' INPUT_IN_HEX='4101999999999999A';

INPUT_CHAR='.2P' INPUT_IN_HEX='4103333333333333';

INPUT_CHAR='.3P' INPUT_IN_HEX='4104CCCCCCCCCCCD';

INPUT_CHAR='.4P' INPUT_IN_HEX='4106666666666666';

THE ELEMENTS OF MATRIX B ARE:

INPUT_CHAR='.5P' INPUT_IN_HEX='4080000000000000';

INPUT_CHAR='.7P' INPUT_IN_HEX='4108333333333333';

INPUT_CHAR='.6P' INPUT_IN_HEX='410999999999999A';

INPUT_CHAR='.8P' INPUT_IN_HEX='410CCCCCCCCCCCCD';

NORMALIZED REPRESENTATION	INTERNAL HEX REPRESENTATION	UNNORMALIZED SIGNIFICANCE REPRESENTATION	SIGNIFICANCE POINTER
C(1)= 1.899999999999999E-01;	41030A3D70A3D70A	1.90000000000000 E-01	18
C(2)= 2.199999999999999E-01;	4103851E8851EB85	2.20000000000000 E-01	18
C(3)= 4.299999999999999E-01;	4106E147AE147AE1	4.30000000000000 E-01	18
C(4)= 5.000000000000000E-01;	4108000000000000	5.00000000000000 E-01	18

MATRIX MULTIPLY USING THE OL2PIVP ROUTINE

N= 3;

THE ELEMENTS OF MATRIX A ARE:

INPUT_CHAR='1P'	INPUT_IN_HEX='4110000000000000';
INPUT_CHAR='2P'	INPUT_IN_HEX='4120000000000000';
INPUT_CHAR='3P'	INPUT_IN_HEX='4130000000000000';
INPUT_CHAR='4.000'	INPUT_IN_HEX='480000000004000';
INPUT_CHAR='5.000'	INPUT_IN_HEX='4A00000000050000';
INPUT_CHAR='6.000'	INPUT_IN_HEX='4A00000000060000';
INPUT_CHAR='7P'	INPUT_IN_HEX='4170000000000000';
INPUT_CHAR='8P'	INPUT_IN_HEX='4180000000000000';
INPUT_CHAR='9P'	INPUT_IN_HEX='4190000000000000';

THE ELEMENTS OF MATRIX B ARE:

INPUT_CHAR='1P'	INPUT_IN_HEX='4110000000000000';
INPUT_CHAR='4.0'	INPUT_IN_HEX='4C000000000000400';
INPUT_CHAR='7P'	INPUT_IN_HEX='4170000000000000';
INPUT_CHAR='2P'	INPUT_IN_HEX='4120000000000000';
INPUT_CHAR='5.00'	INPUT_IN_HEX='480000000005000';
INPUT_CHAR='8P'	INPUT_IN_HEX='4180000000000000';
INPUT_CHAR='3P'	INPUT_IN_HEX='4130000000000000';
INPUT_CHAR='6.000'	INPUT_IN_HEX='4A00000000060000';
INPUT_CHAR='9P'	INPUT_IN_HEX='4190000000000000';

	NORMALIZED REPRESENTATION	INTERNAL HEX REPRESENTATION	UNNORMALIZED SIGNIFICANCE REPRESENTATION		SIGNIFICANCE POINTER
C(1)=	3.000000000000000E+01;	4D00000000001E0	3.0	E+01	5
C(2)=	3.600000000000000E+01;	4C00000000002400	3.600	E+01	7
C(3)=	4.200000000000000E+01;	4B00000000002A000	4.2000	E+01	8
C(4)=	6.600000000000000E+01;	4D00000000000420	6.6	E+01	5
C(5)=	8.100000000000000E+01;	4C00000000005100	8.100	E+01	7
C(6)=	9.600000000000000E+01;	4C00000000006000	9.600	E+01	7
C(7)=	1.020000000000000E+02;	4D00000000000660	1.02	E+02	6
C(8)=	1.260000000000000E+02;	4C00000000007E00	1.2600	E+02	8
C(9)=	1.500000000000000E+02;	4B00000000096000	1.50000	E+02	9

MATRIX MULTIPLY USING THE GL2PIVP ROUTINE

N= 3;

THE ELEMENTS OF MATRIX A ARE:

```
INPUT_CHAR='0P'      INPUT_IN_HEX='0000000000000000';
INPUT_CHAR='0P'      INPUT_IN_HEX='0000000000000000';
INPUT_CHAR='0P'      INPUT_IN_HEX='0000000000000000';
INPUT_CHAR='4.000'   INPUT_IN_HEX='4800000000004000';
INPUT_CHAR='5.000'   INPUT_IN_HEX='4A00000000050000';
INPUT_CHAR='6.000'   INPUT_IN_HEX='4A00000000060000';
INPUT_CHAR='7P'      INPUT_IN_HEX='4170000000000000';
INPUT_CHAR='8P'      INPUT_IN_HEX='4180000000000000';
INPUT_CHAR='9P'      INPUT_IN_HEX='4190000000000000';
```

THE ELEMENTS OF MATRIX B ARE:

```
INPUT_CHAR='1P'      INPUT_IN_HEX='4110000000000000';
INPUT_CHAR='4.0'     INPUT_IN_HEX='4C00000000000400';
INPUT_CHAR='7P'      INPUT_IN_HEX='4170000000000000';
INPUT_CHAR='2P'      INPUT_IN_HEX='4120000000000000';
INPUT_CHAR='5.00'    INPUT_IN_HEX='4B00000000005000';
INPUT_CHAR='8P'      INPUT_IN_HEX='4180000000000000';
INPUT_CHAR='3P'      INPUT_IN_HEX='4130000000000000';
INPUT_CHAR='6.000'   INPUT_IN_HEX='4A00000000060000';
INPUT_CHAR='9P'      INPUT_IN_HEX='4190000000000000';
```

	NORMALIZED REPRESENTATION	INTERNAL HEX REPRESENTATION	UNNORMALIZED SIGNIFICANCE REPRESENTATION		SIGNIFICANCE POINTER
C(1)=	0.0000000000000000E+00;	0000000000000000		0.0P	-3
C(2)=	0.0000000000000000E+00;	0000000000000000		0.0P	-3
C(3)=	0.0000000000000000E+00;	0000000000000000		0.0P	-3
C(4)=	6.6000000000000000E+01;	4D00000000000420	6.6	E+01	5
C(5)=	8.1000000000000000E+01;	4C00000000005100	8.100	E+01	7
C(6)=	9.6000000000000000E+01;	4C00000000006000	9.600	E+01	7
C(7)=	1.0200000000000000E+02;	4D00000000000660	1.02	E+02	6
C(8)=	1.2600000000000000E+02;	4C00000000007E00	1.2600	E+02	8
C(9)=	1.5000000000000000E+02;	4800000000096000	1.50000	E+02	9

```
1                CRAMER: PROC OPTIONS(MAIN);
2                     DCL OL2PILK ENTRY RETURNS ( CHAR(32) VAR );
3                     DCL C CHAR(32) VAR STATIC;
4                     DCL OL2PICF ENTRY (CHAR(32) VAR) RETURNS ( FLOAT BIN(53));
5                     DCL OL2PIFD ENTRY ( FLOAT BIN (53),     /* UNNORMALIZED #   */
                                          FIXED BIN (15),     /*  FIELD WIDTH     */
                                          FIXED BIN (15),     /* FRACTIONAL DIGITS*/
                                          FIXED BIN (15),     /* EORFLAG          */
                                          FIXED BIN (15)  )   /*  UB_NUMBER       */
                                    RETURNS ( CHAR(120) VAR ) ; /* NUMBER IN CHAR   */
6                     DCL (W,D,EORFLAG,UBNUM) FIXED BIN(15);
7                     DCL   NUMBER CHAR(120) VAR;
8                     DCL   @OL2ADD   ENTRY(FLOAT BIN(53),FLOAT BIN(53) )
                                      RETURNS (FLOAT BIN(53));
9                     DCL   @OL2SUB   ENTRY(FLOAT BIN(53),FLOAT BIN(53) )
                                      RETURNS (FLOAT BIN(53));
10                    DCL   @OL2MPY   ENTRY(FLOAT BIN(53),FLOAT BIN(53) )
                                      RETURNS (FLOAT BIN(53));
11                    DCL   @OL2DIV   ENTRY(FLOAT BIN(53),FLOAT BIN(53) )
                                      RETURNS (FLOAT BIN(53));
12                    DCL P(7) FLOAT BIN(53) STATIC;
13                    DCL T(20) FLOAT BIN(53) STATIC;
14                    DCL (HCHAR CHAR(16);
15                    DCL P_IN_HEX CHAR(16);
16                    DCL (NTHEX ENTRY (FLOAT BIN(53)) RETURNS (CHAR(16));
17               ON ENDFILE(SYSIN) GOTO FINIS;
19               ON ERROR  BEGIN;
21               PUT SKIP LIST ('### ERROR HAS OCCURRED ###');
22                    DO I = 1 BY 1 TO 15;
23                    NUMBER = OL2PIFD ( T(I),  W,D,EORFLAG,UBNUM);
24                    IHCHAR = INTHEX( T(I));
25                    PUT SKIP(2);    PUT DATA (T(I));
27                    PUT SKIP(0) EDIT (IHCHAR,NUMBER,UBNUM)
                                       (COL(38),A(16),X(10),A(24),X(10),F(10) );
28                    END;
29               STOP;
30               END;
31                    GET DATA (W,D,EORFLAG);
32                    PUT DATA (W,D,EORFLAG);
33               LOOP: PUT PAGE LIST ('SOLUTION BY CRAMER RULE');
34                    PUT SKIP(2);
35                    DO I = 1 BY 1 TO 6;
                      /* READ IN COEFFICIENTS */
36                    C = OL2PILK;
37                    P(I) = OL2PICF(C);
38                    P_IN_HEX = INTHEX(P(I));
39                    PUT SKIP;
40                    PUT DATA ( I , C , P_IN_HEX);
41                    END;
42                    PUT SKIP(2);
43                    PUT EDIT ('NORMALIZED','INTERNAL HEX',
                                'UNNORMALIZED','SIGNIFICANCE')
```

```
                    (COL(09),A,COL(40),A,COL(68),A,COL(100),A);
44                  PUT EDIT ('REPRESENTATION','REPRESENTATION',
                              'SIGNIFICANCE REPRESENTATION','POINTER')
                    (COL(07),A,COL(39),A,COL(61),A,COL(103),A);
                    /*  SOLUTION OF SYSTEM OF TWO EQUATIONS */
                    /* SYSTEM:  P(1)*X + P(2)*Y = P(3)  */
                    /*          P(4)*X + P(5)*Y = P(6) */
                    /* DENOMINATOR PRODUCTS */
45                  T(1) = @CL2MPY(P(1),P(5));
46                  T(2) = @CL2MPY(P(4),P(2));
                    /* X-NUMERATOR PRODUCTS */
47                  T(3) = @CL2MPY(P(3),P(5));
48                  T(4) = @CL2MPY(P(2),P(6));
                    /* Y-NUMERATOR PRODUCTS */
49                  T(5) = @CL2MPY(P(1),P(6));
50                  T(6) = @OL2MPY(P(3),P(4));
                    /* DENOMINATOR */
51                  T(7) = @OL2SUB(T(1),T(2));
                    /* X-NUMERATOR */
52                  T(8) = @CL2SUB(T(3),T(4));
                    /* Y-NUMERATOR */
53                  T(9) = @OL2SUB(T(5),T(6));
                    /* X-SOLUTION*/
54                  T(10) = @OL2DIV(T(8),T(7));
                    /* Y-SOLUTION*/
55                  T(11) = @OL2DIV(T(9),T(7));
                    /* CALCULATE THE RESIDUAL IN EQN #1 */
56                  T(12) = @OL2MPY(P(1),T(10));
57                  T(13) = @OL2MPY(P(2),T(11));
58                  T(14) =  @CL2ADD(T(12),T(13));
59                  T(15) = @OL2SUB(P(3),T(14));
                    /* CALCULATE THE RESIDUAL IN EQN #2 */
60                  T(16) = @OL2MPY(P(4),T(10));
61                  T(17) = @OL2MPY(P(5),T(11));
62                  T(18) = @OL2ADD(T(16),T(17));
63                  T(19) = @OL2SUB(P(6),T(18));
64                  DO I = 1 BY 1 TO 19;
65                  NUMBER = OL2PIFD ( T(I),  W,D,EORFLAG,U8NUM);
66                  IHCHAR = INTHEX( T(I));
67                  PUT SKIP(2);    PUT DATA (T(I));
69                  PUT SKIP(0) EDIT (IHCHAR,NUMBER,U8NUM)
                                     (COL(38),A(16),X(10),A(24),X(10),F(10) );
70                  END;
71                  GOTO LOOP;
72                  FINIS: PUT SKIP LIST ('END_OF_FILE');
73              END CRAMER;
```

SOLUTION BY CRAMER RULE

```
I=        1          C='1P'                P_IN_HEX='4110000000000000';
I=        2          C='.990000'           P_IN_HEX='480000000FD70A4';
I=        3          C='1.990000'          P_IN_HEX='4800000001FD70A4';
I=        4          C='.990000'           P_IN_HEX='480000000FD70A4';
I=        5          C='.980000'           P_IN_HEX='4800000000FAE148';
I=        6          C='1.970000'          P_IN_HEX='480000001F851EC';
```

	NORMALIZED REPRESENTATION	INTERNAL HEX REPRESENTATION	UNNORMALIZED SIGNIFICANCE REPRESENTATION	SIGNIFICANCE POINTER
T(1)=	9.800000190734862E-01;	480000000FAE148	0.9800000	22
T(2)=	9.801000170409679E-01;	470000000FAE7058	0.98010002	22
T(3)=	1.950200021266937E+00;	4800000001F3404F	1.9502000	22
T(4)=	1.950300037860870E+00;	4800000001F346DD	1.9503000	22
T(5)=	1.970000028610229E+00;	4800000001F851EC	1.9700000	22
T(6)=	1.970100045204162E+00;	4800000001F8587A	1.9701000	22
T(7)=	-9.995698928833007E-05;	C800000000000068D	-0.0001000	22
T(8)=	-1.000165939331054E-04;	C800000000000068E	-0.0001000	22
T(9)=	-1.000165939331054E-04;	C800000000000068E	-0.0001000	22
T(10)=	1.000000000000000E+00;	4C00000000000100	1.00	22
T(11)=	1.000000000000000E+00;	4C00000000000100	1.00	22
T(12)=	1.000000000000000E+00;	4C00000000000100	1.00	22
T(13)=	9.899902343750000E-01;	4800000000000FD7	0.990	22
T(14)=	1.988281250000000E+00;	4C00000000000FD	1.99	22
T(15)=	0.000000000000000E+00;	4C00000000000000	0000000.000000000000000	-3
T(16)=	9.899902343750000E-01;	4800000000000FD7	0.990	22
T(17)=	9.799804687500000E-01;	4800000000000FAE	0.980	22
T(18)=	1.969970703125000E+00;	4800000000001F85	1.970	22
T(19)=	0.000000000000000E+00;	4800000000000000	0000000.000000000000000	-3

SOLUTION BY CRAMER RULE

```
I=      1          C='1P'          P_IN_HEX='411000000000000';
I=      2          C='.99'         P_IN_HEX='480000000000F07';
I=      3          C='1.99'        P_IN_HEX='4C000000000001FD';
I=      4          C='.99'         P_IN_HEX='480000000000F07';
I=      5          C='.98'         P_IN_HEX='480000000000FAE';
I=      6          C='1.97'        P_IN_HEX='4C000000000001F8';
```

NORMALIZED REPRESENTATION	INTERNAL HEX REPRESENTATION	UNNORMALIZED SIGNIFICANCE REPRESENTATION	SIGNIFICANCE POINTER
T(1)= 9.7998046875000000E-01;	480000000000FAE	0.980	22
T(2)= 9.8008728027343750E-01;	4A00C0000000FAE7	0.9801	22
T(3)= 1.9484863281250000E+00;	4B0000000001F20	1.949	22
T(4)= 1.9489746093750000E+00;	4B0000000001F2F	1.949	22
T(5)= 1.9687500000000000E+00;	4C000000000001F8	1.97	22
T(6)= 1.9682617187500000E+00;	4B0000000001F7E	1.968	22
T(7)= 0.0000000000000000E+00;	480000000000000	0000000.000000000000000	-3
T(8)=-4.3828125000000000E-04;	CB0000000000002	-0.001	22
T(9)= 0.0000000000000000E+00;	4C0000000000000	0000000.000000000000000	-3
T(10)= 0.0000000000000000E+00;	4000000000000000	0000000.000000000000000	-3
T(11)= 0.0000000000000000E+00;	4100000000000000	0000000.000000000000000	-3
T(12)= 0.0000000000000000E+00;	4100000000000000	0000000.000000000000000	-3
T(13)= 0.0000000000000000E+00;	4C0000000000000	0000000.000000000000000	-3
T(14)= 0.0000000000000000E+00;	4C0000000000000	0000000.000000000000000	-3
T(15)= 1.9882812500000000E+00;	4C000000000001FD	1.99	22
T(16)= 0.0000000000000000E+00;	480000000000000	0000000.000000000000000	-3
T(17)= 0.0000000000000000E+00;	4C0000000000000	0000000.000000000000000	-3
T(18)= 0.0000000000000000E+00;	4C0000000000000	0000000.000000000000000	-3
T(19)= 1.9687500000000000E+00;	4C000000000001F8	1.97	22

SOLUTION BY CRAMER RULE

```
I=      1          C='0.780P'          P_IN_HEX='410C7AE147AE147B';
I=      2          C='0.563P'          P_IN_HEX='4109020C49BA5E35';
I=      3          C='0.217P'          P_IN_HEX='410378D4FDF38646';
I=      4          C='0.913P'          P_IN_HEX='410E9BA5E353F7CF';
I=      5          C='0.659P'          P_IN_HEX='410A8B4395810625';
I=      6          C='0.254P'          P_IN_HEX='410410624DD2F1AA';
```

	NORMALIZED REPRESENTATION	INTERNAL HEX REPRESENTATION	UNNORMALIZED SIGNIFICANCE REPRESENTATION	SIGNIFICANCE POINTER
T(1)=	5.140200000000001E-01;	4108396D09176B7	0.51402000000000	-1
T(2)=	5.1401900000000C0E-01;	41C8396BFCA85CAB	0.51401900000000	-1
T(3)=	1.4300299999999999E-01;	41C249BD8383AD9F	0.14300300000000	-1
T(4)=	1.4300200000000000E-01;	410249BC77143394	0.14300200000000	-1
T(5)=	1.9812000000000000E-01;	41032B7FE08AEFB3	0.19812000000000	-1
T(6)=	1.9812099999999999E-01;	41032B80ECFA69BE	0.19812100000000	-1
T(7)=	1.0000000000139777E-06;	410J00010C6F7A0C	0.000001000000000	22
T(8)=	9.9999999999177332E-07;	410J00010C6F7A0B	0.000001000000000	22
T(9)=	-9.9999999999177332E-07;	C10J00010C6F7A0B	-0.000001000000000	22
T(10)=	9.999999997671692E-01;	4600000OFFFFFFFF	1.000000000	22
T(11)=	-9.999999997671692E-01;	C60J0000FFFFFFFF	-1.000000000	22
T(12)=	7.799999998242127E-01;	450J000C7AE147A2	0.7799999998	22
T(13)=	-5.629999998636776E-01;	C50J00090C20C49B1	-0.5629999999	22
T(14)=	2.169999999605352E-01;	450J000378D4F0F1	0.2170000000	22
T(15)=	2.910383045673370E-11;	4500C00000000002	0000000.0000000000	-1
T(16)=	9.129999997821868E-01;	450J000E9BA5E345	0.9129999998	22
T(17)=	-6.589999998395796E-01;	C500000A8B439576	-0.6589999998	22
T(18)=	2.539999999426072E-01;	450J00041062400F	0.2539999999	22
T(19)=	4.365574568510055E-11;	450JC0000J000003	0000000.0000000000	-1

SOLUTION BY CRAMER RULE

```
I=        1          C='.7800000000'        P_IN_HEX='440000C7AE147AE1';
I=        2          C='.5630000000'        P_IN_HEX='4400009020C498A6';
I=        3          C='.2170000000'        P_IN_HEX='450000378D4F0F4';
I=        4          C='.9130000000'        P_IN_HEX='440000E98A5E353F';
I=        5          C='.65900C0000'        P_IN_HEX='4400008884395810';
I=        6          C='.2540000000'        P_IN_HEX='4500000410624DD3';
```

NORMALIZED REPRESENTATION	INTERNAL HEX REPRESENTATION	UNNORMALIZED SIGNIFICANCE REPRESENTATION	SIGNIFICANCE POINTER
T(1)= 5.140199999995615E-01;	430J0839600917CF	0.5140199999996	22
T(2)= 5.140189999998483E-01;	430J083968FCA85A	0.5140190000000	22
T(3)= 1.430030000028636E-01;	440J002498D8383E	0.14300300000	22
T(4)= 1.430020000007061E-01;	440J002498C77144	0.14300200000	22
T(5)= 1.981200000092C6E-01;	440J0032B7FE08B0	0.19812000000	22
T(6)= 1.981210000039936E-01;	44000032B80ECFAB	0.19812100000	22
T(7)= 9.999997132581483E-07;	430J0000010C6F75	0.0000009999997	22
T(8)= 1.000002157525159E-06;	440000000010C6FA	0.00000100000	22
T(9)=-1.000003067019861E-06;	C40J000000010C6FB	-0.00000100000	22
T(10)= 1.000002861022949E+00;	4900000000100003	1.00000	22
T(11)=-1.000003814697265E+00;	C900000000100004	-1.00000	22
T(12)= 7.800022363662718E-01;	480JC0000C7AE3A	0.7800022	22
T(13)=-5.630021691322326E-01;	C900000009020E9	-0.5630022	22
T(14)= 2.170000672340393E-01;	480J000000378051	0.2170001	22
T(15)=-5.960464477539062E-08;	C80J000000000001	-0.0000001	22
T(16)= 9.130026102066040E-01;	4800000000E98A8A	0.9130026	22
T(17)=-6.590025424957275E-01;	C80000000A8B464	-0.6590025	22
T(18)= 2.540000677108764E-01;	480JC00000410626	0.2540001	22
T(19)=-5.960464477539062E-08;	C80JC0C000000001	-0.0000001	22

```
1               GAUSS_ELIMINATION: PROC OPTIONS(MAIN);
2                    DCL OL2PILK ENTRY RETURNS ( CHAR(32) VAR );
3                    DCL C CHAR(32) VAR STATIC;
4                    DCL OL2PICF ENTRY (CHAR(32) VAR) RETURNS ( FLOAT BIN(53));
5                    DCL OL2PIFD ENTRY ( FLOAT BIN (53),     /* UNNORMALIZED #   */
                                        FIXED BIN (15),     /*  FIELD WIDTH     */
                                        FIXED BIN (15),     /* FRACTIONAL DIGITS*/
                                        FIXED BIN (15),     /* EORFLAG          */
                                        FIXED BIN (15)  )   /*  UB_NUMBER       */
                                  RETURNS ( CHAR(120) VAR ) ; /* NUMBER IN CHAR   */
6                    DCL (W,D,EORFLAG,UBNUM) FIXED BIN(15);
7                    DCL   NUMBER CHAR(120) VAR;
8                    DCL   @DL2ADD   ENTRY(FLOAT BIN(53),FLOAT BIN(53) )
                                     RETURNS (FLOAT BIN(53));
9                    DCL   @DL2SUB   ENTRY(FLOAT BIN(53),FLOAT BIN(53) )
                                     RETURNS (FLOAT BIN(53));
10                   DCL   @DL2MPY   ENTRY(FLOAT BIN(53),FLOAT BIN(53) )
                                     RETURNS (FLOAT BIN(53));
11                   DCL   @DL2DIV   ENTRY(FLOAT BIN(53),FLOAT BIN(53) )
                                     RETURNS (FLOAT BIN(53));
12                   DCL   PIDIV     ENTRY(FLOAT BIN(53),FLOAT BIN(53) )
                                     RETURNS (FLOAT BIN(53));
13                   DCL P(7) FLOAT BIN(53) STATIC;
14                   DCL T(20) FLOAT BIN(53) STATIC;
15                   DCL IHCHAR CHAR(16);
16                   DCL P_IN_HEX CHAR(16);
17                   DCL INTHEX ENTRY (FLOAT BIN(53)) RETURNS (CHAR(16));
18              ON ENDFILE(SYSIN) GOTO FINIS;
20                   GET DATA (W,D,EORFLAG);
21                   PUT DATA (W,D,EORFLAG);
22              LOOP: PUT PAGE LIST ('SOLUTION BY GAUSS ELIMINATION');
23                   PUT SKIP;
24                   PUT LIST('USING PIDIV WITH NO AUX. RIGHT SHIFTING RULE');
25                   PUT SKIP(2);
                     /* READ IN COEFFICIENTS */
26                   DO I = 1 BY 1 TO 6;
27                   C = OL2PILK;
28                   P(I) = OL2PICF(C);
29                   P_IN_HEX = INTHEX(P(I));
30                   PUT SKIP;
31                   PUT DATA ( I , C , P_IN_HEX);
32                   END;
33                   PUT SKIP(2);
34                   PUT EDIT ('NORMALIZED','INTERNAL HEX',
                               'UNNORMALIZED','SIGNIFICANCE')
                              (COL(09),A,COL(40),A,COL(68),A,COL(100),A);
35                   PUT EDIT ('REPRESENTATION','REPRESENTATION',
                               'SIGNIFICANCE REPRESENTATION','POINTER')
                              (COL(07),A,COL(39),A,COL(61),A,COL(103),A);
                /* P(1) IS ASSUMED TO BE 1.0P */
                     /* SYSTEM:  P(1)*X + P(2)*Y = P(3)  */
                     /*          P(4)*X + P(5)*Y = P(6) */
```

```
36                  T(1) =   PIDIV(P(6),P(4));
37                  T(2) = @OL2SUB(P(3),T(1));
38                  T(3) =   PIDIV(P(5),P(4));
39                  T(4) = @OL2SUB(P(2),T(3));
40                  T(5) =   PIDIV(T(2),T(4));    /* T(5) IS Y-SOLUTION. */
41                  T(6) =@OL2MPY(P(2),T(5));
42                  T(7) = @OL2SUB(P(3),T(6));    /* T(7) IS X-SOLUTION. */
43                  T(8) = @OL2MPY(P(1),T(7));
44                  T(9) = @OL2MPY(P(2),T(5));
45                  T(10) = @OL2ADD(T(8),T(9));
46                  T(11) = @OL2SUB(P(3),T(10)); /* T(11) IS EQN #1 RESIDUAL */
47                  T(12) = @OL2MPY (P(4),T(7));
48                  T(13) = @OL2MPY(P(5),T(5));
49                  T(14) = @OL2ADD(T(12),T(13));
50                  T(15) = @OL2SUB(P(6),T(14)); /* T(15) IS EQN #2 RESIDUAL */
51                  DO I = 1 BY 1 TO 15;
52                  NUMBER = OL2PIFO ( T(I),  W,D,EORFLAG,UBNUM);
53                  IHCHAR = (NTHEX( T(I));
54                  PUT SKIP(2);    PUT DATA (T(I));
56                  PUT SKIP(0) EDIT (IHCHAR,NUMBER,UBNUM)
                                 (COL(38),A(16),X(10),A(24),X(10),F(10) );
57                  END;
58                  GOTO LOOP;
59                  FINIS: PUT SKIP LIST ('END_OF_FILE');
60              END GAUSS_ELIMINATION;
```

SOLUTION BY GAUSS ELIMINATION
USING PIDIV WITH NO AUX. RIGHT SHIFTING RULE

```
I=        1          C='1.0P'          P_IN_HEX='4110000000000000';
I=        2          C='0.563P'        P_IN_HEX='410902DC498A5E35';
I=        3          C='0.217P'        P_IN_HEX='410378D4FDF38646';
I=        4          C='0.913P'        P_IN_HEX='41DE98A5E353F7CF';
I=        5          C='0.659P'        P_IN_HEX='410A884395810625';
I=        6          C='0.254P'        P_IN_HEX='410410624DD2F1AA';
```

	NORMALIZED REPRESENTATION	INTERNAL HEX REPRESENTATION	UNNORMALIZED SIGNIFICANCE REPRESENTATION	SIGNIFICANCE POINTER
T(1)=	2.782037239868564E-01;	41047385BF826223	0.27820372398685	-1
T(2)=	-6.120372398685635E-02;	C100FA8DC18EABDD	-0.061203723986856	22
T(3)=	7.217962760131435E-01;	41088C7A407D9D00	0.72179627601314	-1
T(4)=	-1.587962760131436E-01;	C1028A6DF6C33FA8	-0.15879627601314	-1
T(5)=	3.854229174857373E-01;	420062AB13898AE8	0.38542291748574	22
T(6)=	2.169931025444700E-01;	410378CDC26E2D58	0.21699310254447	-1
T(7)=	6.897455530063822E-06;	4100D0D73B8588E8	0.000006897455530	22
T(8)=	6.897455530063822E-06;	4100D007388588E8	0.000006897455530	22
T(9)=	2.169931025444700E-01;	410378CDC26E2D58	0.21699310254447	-1
T(10)=	2.170000000000C00E-01;	410378D4FDF38646	0.21700000000000	-1
T(11)=	0.000000000000000E+00;	4100000000000000	.0000000.000000000000000	-3
T(12)=	6.297376898944939E-06;	400D0069A7071FF8	0.0000062973768989	22
T(13)=	2.539937026231009E-01;	4104105883627FAA	0.25399370262310	-1
T(14)=	2.539999999999997E-01;	410410624002F1A9	0.25400000000000	-1
T(15)=	2.220446049250313E-16;	410DC0000D000001	0000000.00000000000000	-1

SOLUTION BY GAUSS ELIMINATION
USING PIDIV WITH NO AUX. RIGHT SHIFTING RULE

```
I=        1          C='1.0P'          P_IN_HEX='4110000000000000';
I=        2          C='0.563'         P_IN_HEX='4A0000000009021';
I=        3          C='0.217'         P_IN_HEX='4B0000000000379';
I=        4          C='0.913'         P_IN_HEX='4A000000000E9BA';
I=        5          C='0.659'         P_IN_HEX='4A0000000000A884';
I=        6          C='0.254'         P_IN_HEX='4B0000000000410';
```

	NORMALIZED REPRESENTATION	INTERNAL HEX REPRESENTATION	UNNORMALIZED SIGNIFICANCE REPRESENTATION	SIGNIFICANCE POINTER
T(1)=	2.780761718750000E-01;	4B0J000000000473	0.278	22
T(2)=	-6.103515625000000E-02;	CB0J000000000FA	-0.061	22
T(3)=	7.218017578125000E-01;	4A000000000088C8	0.7218	22
T(4)=	-1.587982177734375E-01;	CA0J000000000287	-0.1588	22
T(5)=	3.828125000000000E-01;	4C000000000000062	0.38	22
T(6)=	2.155761718750000E-01;	4B0J000000000373	0.216	22
T(7)=	1.464843750000000E-03;	4B0J000000000006	0.001	22
T(8)=	1.464843750000000E-03;	4B0J000000000006	0.001	22
T(9)=	2.155761718750000E-01;	4B000000000000373	0.216	22
T(10)=	2.170410156250000E-01;	4B0JC00000000379	0.217	22
T(11)=	0.000000000000000E+00;	4B0J000000000000	0000000.000000000000000	-3
T(12)=	1.342773437500000E-03;	4A000000000000058	0.0013	22
T(13)=	2.521972656250000E-01;	4B000000000000409	0.252	22
T(14)=	2.534179687500000E-01;	4B000000000000040E	0.253	22
T(15)=	4.882812500000000E-04;	4B0J000000000002	0.001	22

SOLUTION BY GAUSS ELIMINATION
USING PIOIV WITH NO AUX. RIGHT SHIFTING RULE

```
I=        1          C='1P'              P_IN_HEX='4110000000000000';
I=        2          C='.99P'            P_IN_HEX='410FD70A3070A307';
I=        3          C='1.99P'           P_IN_HEX='4201FD70A3D70A30';
I=        4          C='.99P'            P_IN_HEX='410F070A3070A3D7';
I=        5          C='.98P'            P_IN_HEX='410FAE147AE147AE';
I=        6          C='1.97P'           P_IN_HEX='4201F851EB851EB8';
```

	NORMALIZED REPRESENTATION	INTERNAL HEX REPRESENTATION	UNNORMALIZED SIGNIFICANCE REPRESENTATION	SIGNIFICANCE POINTER
T(1)=	1.985898989898989E+00;	4201F06A0528F5A8	1.98989898989899	-1
T(2)=	1.010101010088249E-04;	420000069EAB1495	0.00010101010101	22
T(3)=	9.898989898989858E-01;	410FD6A0528F5A81	0.98989898989899	-1
T(4)=	1.010101010101571E-04;	41000069EAB14956	0.0001010101010101	22
T(5)=	9.999999999854479E-01;	450J000FFFFFFFFF	1.0000000000	22
T(6)=	9.899999999852297E-01;	440000F070A306FA	0.98999999999	22
T(7)=	1.0C0000000014551E+00;	440JC10000000010	1.00000000001	22
T(8)=	1.000000000014551E+00;	440J010000000010	1.00000000001	22
T(9)=	9.899999999852297E-01;	440000FD70A3D6FA	0.98999999999	22
T(10)=	1.989999999999781E+00;	440J01FD70A3070A	1.99000000000	22
T(11)=	0.000000000000000E+00;	4400000000000000	0000000.000000000000000	-3
T(12)=	9.900000000143904E-01;	430J0FD70A3D71A1	0.9900000000145	22
T(13)=	9.799999999859209E-01;	440J00FAE147AE05	0.97999999999	22
T(14)=	1.970000000000254E+00;	440J01F851EB851F	1.97000000000	22
T(15)=	0.000000000000000E+00;	440J000000000000	0000000.000000000000000	-3

SOLUTION BY GAUSS ELIMINATION
USING PIDIV WITH NO AUX. RIGHT SHIFTING RULE

```
I=      1          C='1P'                P_IN_HEX='411000000000000';
I=      2          C='.990000'           P_IN_HEX='4800000000FD70A4';
I=      3          C='1.990000'          P_IN_HEX='4800000001FD70A4';
I=      4          C='.990000'           P_IN_HEX='4800000000FD70A4';
I=      5          C='.980000'           P_IN_HEX='4800000000FAE148';
I=      6          C='1.970000'          P_IN_HEX='4800000001F851EC';
```

	NORMALIZED REPRESENTATION	INTERNAL HEX REPRESENTATION	UNNORMALIZED SIGNIFICANCE REPRESENTATION	SIGNIFICANCE POINTER
T(1)=	1.989898681640625E+00;	490000000001FD6A0	1.98990	22
T(2)=	1.01C894775390625E-04;	490JC00000000D6A	0.00010	22
T(3)=	9.898989796638488E-01;	480J000000FD6A05	0.9898990	22
T(4)=	1.010298728942871E-04;	480JC00000J00069F	0.0001010	22
T(5)=	1.000000000000000E+00;	4D00C000000000010	1.00000000000000	7
T(6)=	9.882812500000C0E-01;	4C000000000000FD	0.99	22
T(7)=	1.00C000000000J000E+00;	4C0JC00000000100	1.00	22
T(8)=	1.000000000000000E+00;	4C0J000000000100	1.00	22
T(9)=	9.882812500000C0E-01;	4C0J000000000FD	0.99	22
T(10)=	1.988281250000J0CE+00;	4C0J000000001FD	1.99	22
T(11)=	0.000000000000000E+00;	4C0JC00000J000000	0000000.000000000000000	-3
T(12)=	9.899902343750000E-01;	480J000000000FD7	0.990	22
T(13)=	9.804687500000000E-01;	4C0J000000J0000FB	0.98	22
T(14)=	1.968750000000C00E+00;	4C0J000000J0001F8	1.97	22
T(15)=	0.000000000000000E+00;	4C0J00C000000J00	0000000.000000000000000	-3

SOLUTION BY GAUSS ELIMINATION
USING PIOIV WITH NO AUX. RIGHT SHIFTING RULE

```
I=        1          C='1P'              P_IN_HEX='411000000000000';
I=        2          C='.99'             P_IN_HEX='4B0000000000F07';
I=        3          C='1.99'            P_IN_HEX='4C000000000001F0';
I=        4          C='.99'             P_IN_HEX='4B0000000000F07';
I=        5          C='.98'             P_IN_HEX='4B0000000000FAE';
I=        6          C='1.97'            P_IN_HEX='4C000000000001F8';
```

	NORMALIZED REPRESENTATION	INTERNAL HEX REPRESENTATION	UNNORMALIZED SIGNIFICANCE REPRESENTATION	SIGNIFICANCE POINTER
T(1)=	1.988281250000000E+00;	4C000000000001FD	1.99	22
T(2)=	0.000000000000000E+00;	4C0000000000000	0000000.000000000000000	-3
T(3)=	9.899902343750000E-01;	4B0000000000F07	0.990	22
T(4)=	0.000000000000000E+00;	4B0000000000000	0000000.000000000000000	-3
T(5)=	0.000000000000000E+00;	4100000000000000	0000000.000000000000000	-3
T(6)=	0.000000000000000E+00;	4C0000000000000	0000000.000000000000000	-3
T(7)=	1.988281250000000E+00;	4C000000000001F0	1.99	22
T(8)=	1.988281250000000E+00;	4C000000000001F0	1.99	22
T(9)=	0.000000000000000E+00;	4C0000000000000	0000000.000000000000000	-3
T(10)=	1.988281250000000E+00;	4C000000000001F0	1.99	22
T(11)=	0.000000000000000E+00;	4C0000000000000	0000000.000000000000000	-3
T(12)=	1.968261718750000E+00;	4B0000000001F7E	1.968	22
T(13)=	0.000000000000000E+00;	4C0000000000000	0000000.000000000000000	-3
T(14)=	1.964843750000000E+00;	4C000000000001F7	1.97	22
T(15)=	3.906250000000000E-03;	4C0000000000001	0000000.00	-1

APPENDIX B

EXAMPLES OF DOCUMENTATION OF THE BASIC ROUTINES

APPENDIX B

This appendix contains the assembly listings of several significant digit routines: @OL2ADD, @OL2MPY, and OL2PIVP. One of the assembly language macros used to generate the signigicant digit function is also included. It should be noted that in its present form, OL2PIVP (inner product) assumes the elements in the input vector occupying consecutive locations. The indexing scheme can be modified to handle more complicated storage schemes as found in the array structures of OL/2 language (20).

```
  LOC  OBJECT CODE     A00R1 A00R2  STMT   SOURCE STATEMENT                                          22 JUN 72
                                      2 *   OCL @0L2A00 ENTRY ( FLOAT BIN(53), FLOAT BIN(53) )       00000200
                                      3 *               RETURNS ( FLOAT BIN(53) );                   00000300
000000                                4 @0L2A00  CSECT                                               00000400
                                      5         ENTRY @0L2SUB            ANOTHER ENTRY POINT         00000500
000000                                6         USING *,15                                           00000600
000000 47F0 F00C            000C      7         B     *+12                                           00000700
000004 07                             8         OC    AL1(7)                                         00000800
000005 7C0603F2C1C4C4                 9         OC    CL7'@0L2A00'                                   00000900
00000C 9012 0018            0018     10         STM   R1,R2,24(R13)      SAVE REGISTERS              00001000
000010 5821 0004            0004     11         L     R2,4(R1)           R2 -> 2NO OPERAND           00001100
000014 6820 2000            0000     12         LD    F2,0(0,R2)                                     00001200
000018 45F0 F036            0036     13         BAL   R15,AMERGE         SET BASE REGISTER -> @0L2SUB 00001300
                                     14         DROP  15                                             00001400
00001C                               15 @0L2SUB  EQU   *                                            00001500
00001C                               16         USING @0L2SUB,15         ESTABLISH BASE REGISTER     00001600
00001C 47F0 F00C            002B     17         B     *+12                                           00001700
000020 07                            18         OC    AL1(7)                                         00001800
000021 7C0603F2E2E4C2                19         OC    CL7'@0L2SUB'                                   00001900
000028 9012 0018            0018     20         STM   R1,R2,24(R13)      SAVE REGISTERS              00002000
00002C 5821 0004            0004     21         L     R2,4(R1)           GET ADR OF 2ND OPERAND      00002100
000030 6820 2000            0000     22         LD    F2,0(0,R2)         GET 2ND OPERAND             00002200
000034 2322                          23         LCDR  F2,F2              COMPLEMENT 2ND OPERAND      00002300
000036 5811 0000            0000     24 AMERGE  L     R1,0(R1)           R1 -> 1ST OPERAND           00002400
00003A 6801 C000            0000     25         LD    F0,0(R1)           GET 1ST OPERAND             00002500
00003E 2222                          26         LTDR  F2,F2              CHECK FOR ZERO FRACTION     00002600
000040 4780 F10A            0126     27         BZ    A2OPZERO                                       00002700
000044 2200                          28         LTDR  F0,F0              CHECK FOR ZERO FRACTION     00002800
000046 4780 F12B            0144     29         BZ    A1OPZERO                                       00002900
                                     30 *   DETERMINE IF OPERANDS ARE PRECISE                       00003000
00004A 91F0 1001      00001          31         TM    1(R1),X'F0'        TEST 1ST FRACTION           00003100
00004E 4780 F03E            005A     32         BZ    AIMPREC                                        00003200
000052 91F0 2001      00001          33         TM    1(R2),X'F0'        TEST 2ND FRACTION           00003300
000056 4770 F08E            00AA     34         BNZ   APRECISE                                       00003400

00005A 2E02                          36 AIMPREC  AWR   F0,F2              PERFORM SIGNIFICANT A00/SUB 00003600
00005C 4780 FC66            00B2     37         BZ    AZEROFIX           CHECK FOR ZERO FRACTION     00003700
000060 6000 F194            01B0     38         STD   F0,AWORK                                       00003800
000064 91F0 F195      001B1          39         TM    AWORK+1,X'F0'      DID A00 CARRY INTO LEADING ZERO? 00003900
000068 4780 F05B            0074     40         BZ    ARETURN            NO, THE RESULT IS STILL IMPRECISE 00004000
00006C 9035 0020            0020     41         STM   R3,R5,32(R13)      SAVE REGS FOR AUNNORM       00004100
000070 47F0 F0E4            0100     42         B     AUNNORM2           GO MAKE RESULT IMPRECISE    00004200

000074 9812 001B            001B     44 ARETURN  LM    R1,R2,24(R13)      RESTORE R1 AND R2           00004400
000078 5811 C008            0008     45         L     R1,8(R1)           GET PL/1 FUNCTION RETURN ADR 00004500
00007C 6001 0000            0000     46         STD   F0,0(R1)           STORE PL/1 RETURN VALUE     00004600
000080 07FE                          47         BR    R14                RETURN IMPRECISE SUM/DIFF   00004700

000082 9035 0020            0020     49 AZEROFIX STM   R3,R5,32(R13)      CASE OF RELATIVE ZERO RESULT 00004900
000086 4100 007F            007F     50         LA    R0,X'7F'                                       00005000
00008A                               51 AZEROF05 EQU   *                ENTRY FROM AIMPZERO          00005100
00008A 4331 0C00            0000     52         IC    R3,0(R1)           GET 1ST EXP                 00005200
00008E 1430                          53         NR    R3,R0              REMOVE FRACTION SIGN BIT    00005300
000090 4342 0C00            0000     54         IC    R4,0(R2)           GET 2ND EXP                 00005400
000094 1440                          55         NR    R4,R0                                          00005500
000096 1934                          56         CR    R3,R4              WANT THE MAX OF TWO EXPONENTS 00005600
```

LOC	OBJECT CODE	ADDR1	ADDR2	STMT	SOURCE STATEMENT			22 JUN 72
000098	4780 F082		00095	57		BNL	AZEROF10	00005700
00009C	1834			58		LR	R3,R4	00005800
00009E	4230 F18C		001A8	59	AZEROF10	STC	R3,AZERO	00005900
0000A2	6800 F18C		001A8	60		LD	F0,AZERO	00006000
0000A6	47F0 F0FC		00118	61		B	APEXIT	RETURN WITH RELATIVE ZERO 00006100
0000AA	2A02			63	APRECISE	ADR	F0,F2	PERFORM PRECISE ADD/SUB 00006300
0000AC	4770 F0A2		0008E	64		BNZ	APCKEXP	SKIP IF NOT ZERO FRACTION RESULT 00006400
				65	*	TEST FOR TRUE CASE FROM EQUAL		PRECISE OPERANDS 00006500
0000B0	D506 1001 2001 00001		00001	66		CLC	1(7,R1),1(R2)	CHECK NORMALIZED FRACTIONS 00006600
0000B6	4770 F066		00082	67		BNE	AZEROFIX	00006700
0000BA	47F0 F058		00074	68		B	ARETURN	00006800
0000BE				70	APCKEXP	EQU	* CHECK FOR LOST OF PRECISION DUE TO SHIFTING FRACT	00007000
0000BE	9035 0020		00020	71		STM	R3,R5,32(R13)	SAVE MORE REGS 00007100
0000C2	4150 007F		0007F	72		LA	R5,X'7F'	MASK TO REMOVE SIGN BIT 00007200
0000C6	4331 0000		00000	73		IC	R3,0(R1)	GET 1ST EXPONENT 00007300
0000CA	1435			74		NR	R3,R5	REMOVE SIGN BIT 00007400
0000CC	4342 0000		00000	75		IC	R4,0(R2)	GET 2ND EXPONENT 00007500
0000D0	1445			76		NR	R4,R5	REMOVE SIGN BIT 00007600
0000D2	1834			77		SR	R3,R4	R3 := DIFFERENCE OF EXPONENTS 00007700
0000D4	4780 F0FC		00118	78		BZ	APEXIT	NO SHIFTING WAS DONE 00007800
0000D8	1841			79		LR	R4,R1	R4 HAS ADR OF LARGER OPERAND 00007900
0000DA	1852			80		LR	R5,R2	R5 HAS ADR OF SMALLER OPERAND 00008000
0000DC	4720 F0CA		000E6	81		BP	AP01	BRANCH, IF R4 AND R5 ARE CORRECT 00008100
0000E0	1842			82		LR	R4,R2	00008200
0000E2	1851			83		LR	R5,R1	00008300
0000E4	1033			84		LPR	R3,R3	WANT ABS DIFF OF EXPONENTS 00008400
0000E6	5930 F184		001A0	85	AP01	C	R3,=F'14'	WAS THERE SHIFTING PAST GUARD DIGIT 00008500
0000EA	4780 F0E0		000FC	86		BNL	AUNNORM	IF SO, MAKE RESULT IMPRECISE 00008600
0000EE	2820			87		LDR	F2,F0	COPY RESULT 00008700
0000F0	6824 0000		00000	88		SD	F2,0(R4)	SUB WITH NO SHIFTING 00008800
0000F4	6925 0000		00000	89		CD	F2,0(R5)	COMPARE WITH SMALLER OPERAND 00008900
0000F8	4780 F0FC		00118	90		BE	APEXIT	IF EQUAL, RETURN WITH PRECISE RESULT 00009000
0000FC	6000 F194		00180	92	AUNNORM	STD	F0,AWORK	MAKE RESULT IMPRECISE WITH ROUNDING 00009200
000100	4330 F194		001B0	93	AUNNORM2	IC	R3,AWORK	GET RESULTANT EXPONENT 00009300
000104	4230 F19C		00188	94		STC	R3,AINCR	FIX EXPONENT FOR ROUND UP 00009400
000108	6A00 F19C		00188	95		AD	F0,AINCR	ROUND UP 00009500
00010C	4133 0001		00001	96		LA	R3,1(R3)	UP EXPONENT TO SHIFT FRACTION RIGHT 00009600
000110	4230 F18C		001A8	97		STC	R3,AZERO	STORE EXPONENT FOR UNNORMALIZATION 00009700
000114	6E00 F18C		001A8	98		AW	F0,AZERO	UNNORMALIZE RESULT 00009800
000118	9815 0018		00018	100	APEXIT	LM	R1,R5,24(R13)	RESTORE R1 THRU R5 00010000
00011C	5811 0008		00008	101		L	R1,8(R1)	GET PL/1 FUNCTION RETURN ADR 00010100
000120	6001 0000		00000	102		STD	F0,0(R1)	STORE PL/1 RETURN VALUE 00010200
000124	07FE			103		BR	R14	RETURN 00010300
				105	*	SECOND OPERAND IS A ZERO. FIRST OPERAND HAS NOT BEEN CHECKED.		00010500
000126	9500 2000	00000		106	A2OPZERO	CLI	0(R2),X'00'	CHECK FOR TRUE ZERO 00010600
00012A	4780 F058		00074	107		BE	ARETURN	YES, RETURN WITH ADDITIVE IDENTITY 00010700
00012E	2200			108		LTDR	F0,F0	NO, CHECK 1ST OPERAND FOR ZERO 00010800
000130	4770 F13A		00156	109		BNZ	AIMPZERO	BRANCH, IF NON-ZERO + RZ 00010900
000134	9500 1000	00000		110		CLI	0(R1),X'00'	IS 1ST OPERAND A TRUE ZERO 00011000
000138	4770 F066		00082	111		BNE	AZEROFIX	BRANCH, IF RZ + RZ 00011100

```
  LOC  OBJECT CODE     ADDR1 ADDR2  STMT   SOURCE STATEMENT                                                22 JUN 72
000138 6802 0000            00000   112          LD    F0,0(R2)                                                00011200
00013C 47F0 F058            00074   113          B     ARETURN             CASE OF TZ + RZ                     00011300

                                    115 *   SECOND OPERAND WAS NOT ZERO BUT FIRST OPERAND IS A ZERO.            00011500
000144 9500 1000      00000         116 A1OPZERO CLI   0(R1),X'00'         CHECK 1ST OPERAND FOR TRUE ZERO     00011600
000148 6802 0000            00000   117          LD    F0,0(R2)            LOAD NON-ZERO                       00011700
00014C 4780 F058            00074   118          BE    ARETURN             YES, RETURN WITH ADDITIVE IDENTITY  00011800
000150 1801                         119          LR    R0,R1                                                   00011900
000152 1812                         120          LR    R1,R2               R1 -> NON-ZERO FRACTION             00012000
000154 1820                         121          LR    R2,R0               R2 -> RELATIVE ZERO                 00012100
000156                              122 AIMPZERO EQU   *   CASE OF RELATIVE ZERO WITH NON-ZERO                00012200
000156 9C35 0020            00020   123          STM   R3,R5,32(R13)       SAVE MORE REGS                      00012300
00015A 4100 007F            0007F   124          LA    R0,X'7F'            REMOVE SIGN BIT                     00012400
00015E 4331 0000            00000   125          IC    R3,0(R1)            GET EXP OF NON-ZERO FRACTION        00012500
000162 1430                         126          NR    R3,R0                                                   00012600
000164 4342 0000            00000   127          IC    R4,0(R2)            GET EXP OF RELATIVE ZERO            00012700
000168 1440                         128          NR    R4,R0                                                   00012800
00016A 6801 0000            00000   129          LD    F0,0(R1)            LOAD NON-ZERO NUMBER                00012900
00016E 1834                         130          SR    R3,R4               IS EXP(NON-ZERO) .GE. EXP(RZ)?      00013000
000170 4740 F164            00180   131          BM    AIMPZ10             NO, BRANCH                          00013100
000174 91F0 1001      00001         132          TM    1(R1),X'F0'         TEST NON-ZERO FOR PRECISE           00013200
000178 4770 F0E0            000FC   133          BNZ   AUNNORM             YES, MAKE NON-ZERO IMPRECISE        00013300
00017C 47F0 F0FC            00118   134          B     APEXIT              NO, RETURN WITH IMPRECISE NON-ZERO  00013400
000180 1033                         135 AIMPZ10  LPR   R3,R3          CASE OF EXP(NON-ZERO) .LT. EXP(RZ)      00013500
000182 5930 F184            001A0   136          C     R3,=F'14'           SHIFTING PAST GUARD DIGITS?         00013600
000186 4780 F06E            000C8A  137          BNL   AZEROF05            YES, RETURN WITH RELATIVE ZERO      00013700
00018A 6000 F194            00180   138          STD   F0,AWORK                                                00013800
00018E 4340 F194            00180   139          IC    R4,AWORK                                                00013900
000192 1A43                         140          AR    R4,R3                                                   00014000
000194 4240 F18C            001A8   141          STC   R4,AZERO                                                00014100
000198 6E00 F18C            001A8   142          AW    F0,AZERO            SHIFT RIGHT BY THE EXP DIFF         00014200
00019C 47F0 F0FC            00118   143          B     APEXIT                                                  00014300
0001A0                              144          LTORG                                                         00014400
0001A0 0000000E                     145               =F'14'
0001A4 00000000
0001A8 0000000000000000             146 AZERO    DC    D'0'                ZERO FRACTION CONSTANT              00014500
000180                              147 AWORK    DS    D                   WORK AREA FOR RESULT                00014600
000188 FF00000000000008             148 AINCR    DC    X'FF00000000000008'     ROUNDING FRACTION CONSTANT      00014700
                                    149          DROP  15                                                      00014800
                                    150 *   REGISTER DEFINITIONS                                              00014900
000000                              151 R0       EQU   0                                                       00015000
000001                              152 R1       EQU   1                                                       00015100
000002                              153 R2       EQU   2                                                       00015200
000003                              154 R3       EQU   3                                                       00015300
000004                              155 R4       EQU   4                                                       00015400
000005                              156 R5       EQU   5                                                       00015500
00000D                              157 R13      EQU   13                                                      00015600
00000E                              158 R14      EQU   14                                                      00015700
00000F                              159 R15      EQU   15                                                      00015800
000000                              160 F0       EQU   0                                                       00015900
000002                              161 F2       EQU   2                                                       00016000
000004                              162 F4       EQU   4                                                       00016100
000006                              163 F6       EQU   6                                                       00016200
                                    164          END                                                           00016300
```

```
 LOC  OBJECT CODE     ADDR1 ADDR2  STMT   SOURCE STATEMENT                                  21 JUN 72

                                     2 *   DCL @OL2MPY ENTRY ( FLOAT BIN(53), FLOAT BIN(53) )      00000200
                                     3 *             RETURNS ( FLOAT BIN(53) );                    00000300
000000                               4 @OL2MPY  CSECT                                              00000400
000000                               5          USING *,15            ESTABLISH BASE REGISTER      00000500
000000 47F0 F00C           0000C     6          B     *+12                                         00000600
000004 07                            7          DC    AL1(7)                                       00000700
000005 7CD6D3F2D4D7E8                8          DC    CL7'@OL2MPY'                                 00000800
00000C 9016 D018           00018     9          STM   R1,R6,24(R13)   SAVE REGISTERS              00000900
000010 5821 0004           00004    10          L     R2,4(R1)        R2 -> SECOND OPERAND         00001000
000014 5811 0000           00000    11          L     R1,0(R1)        R1 -> FIRST OPERAND          00001100
000018 4100 007F           0007F    12          LA    R0,X'7F'        R0 HAS MASK FOR EXPONENT FIELD 00001200
                                    13 *   CHECK FOR ZERO FACTORS                                  00001300
00001C 6800 1000           00000    14          LD    F0,0(0,R1)      GET MULTIPLICAND             00001400
000020 2200                         15          LTDR  F0,F0                                        00001500
000022 4780 F18A           0018A    16          BZ    MZPROD1                                      00001600
000026 6820 2000           00000    17          LD    F2,0(0,R2)      GET MULTIPLIER               00001700
00002A 2222                         18          LTDR  F2,F2                                        00001800
00002C 4780 F1C2           001C2    19          BZ    MZPROD2                                      00001900
                                    20 *   TEST FOR PRECISE OPERANDS                               00002000
000030 91F0 1001      00001         21          TM    1(R1),X'F0'     TEST FRACTION                00002100
000034 4780 F040           00040    22          BZ    MIMPREC                                      00002200
000038 91F0 2001      00001         23          TM    1(R2),X'F0'     TEST FRACTION                00002300
00003C 4770 F118           00118    24          BNZ   MPRECISE                                     00002400

                                    26 *   IMPRECISE SECTION                                       00002600
                                    27 *   M REPRESENTS THE NUMBER OF SIGNIFICANT HEX ZEROS IN FRACTION 00002700
000040 4160 000E           0000E    28 MIMPREC LA    R6,14                                         00002800
000044 4331 0000           00000    29          IC    R3,0(R1)                                     00002900
000048 1430                         30          NR    R3,R0           R3 := E1                     00003000
00004A 1936                         31          CR    R3,R6           CHECK FOR POSSIBLE EXP UNDERFLOW 00003100
00004C 4780 F060           00060    32          BNL   MOP1OK          BRANCH, IF EXP .GE. 14       00003200
000050 4133 000E           0000E    33          LA    R3,14(R3)                                    00003300
000054 6000 F200           00200    34          STD   F0,WORK1                                     00003400
000058 4230 F200           00200    35          STC   R3,WORK1        USE DIFFERENT EXPONENT FIELD 00003500
00005C 6800 F200           00200    36          LD    F0,WORK1                                     00003600
000060 6A00 F218           00218    37 MOP1OK  AD    F0,TZERO        NORMALIZE TO FIND M1         00003700
000064 6000 F200           00200    38          STD   F0,WORK1                                     00003800
000068 4340 F200           00200    39          IC    R4,WORK1                                     00003900
00006C 1440                         40          NR    R4,R0           R4 := NORMALIZED EXPONENT    00004000
00006E 1834                         41          SR    R3,R4           R3 := M1                     00004100
000070 4352 0000           00000    42          IC    R5,0(R2)                                     00004200
000074 1450                         43          NR    R5,R0           R5 := E2                     00004300
000076 1956                         44          CR    R5,R6           CHECK FOR POSSIBLE EXP UNDERFLOW 00004400
000078 4780 F08C           0008C    45          BNL   MOP2OK          BRANCH, IF EXP .GE. 14       00004500
00007C 4155 000E           0000E    46          LA    R5,14(R5)                                    00004600
000080 6020 F208           00208    47          STD   F2,WORK2                                     00004700
000084 4250 F208           00208    48          STC   R5,WORK2        USE DIFFERENT EXPONENT FIELD 00004800
000088 6820 F208           00208    49          LD    F2,WORK2                                     00004900
00008C 6A20 F218           00218    50 MOP2OK  AD    F2,TZERO        NORMALIZE TO FIND M2         00005000
000090 6020 F208           00208    51          STD   F2,WORK2                                     00005100
000094 4360 F208           00208    52          IC    R6,WORK2                                     00005200
000098 1460                         53          NR    R6,R0           R6 := NORMALIZED EXPONENT    00005300
00009A 1B56                         54          SR    R5,R6           R5 := M2                     00005400

00009C 6800 1000           00000    56          LD    F0,0(0,R1)      GET MULTIPLICAND             00005600
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                  21 JUN 72
0000A0 6C00 2000           00000    57          MD    F0,0(0,R2)        DO NORMALIZED MULTIPLY          00005700
0000A4 2200                         58          LTDR  F0,F0             TEST FOR ZERO FRACTION          00005800
0000A6 4780 F1D0           001D0    59          BZ    MZEROFIX                                          00005900
0000AA 6000 F210           00210    60          STD   F0,WORK3                                          00006000

0000AE 1935                         62 GETMAXM  CR    R3,R5                                             00006200
0000B0 4780 F0B6           000B6    63          BNL   MAUXRULE                                          00006300
0000B4 1835                         64          LR    R3,R5             R3 := MAX(M1,M2)                00006400

0000B6 4100 007F           0007F    66 MAUXRULE LA    R0,X'7F'          RESTORE MASK FOR EXPONENT       00006600
0000BA 1A46                         67          AR    R4,R6       AUX RULE: IF E3 = E1+E2, THEN M3 = M3+1 00006700
0000BC 5B40 F238           00238    68          S     R4,=F'64'         CORRECT FOR EXCESS-64 AOO       00006800
0000C0 4360 F210           00210    69          IC    R6,WORK3                                          00006900
0000C4 1460                         70          NR    R6,R0             R6 := E3                        00007000
0000C6 1964                         71          CR    R6,R4             E3 : (E1+E2)                    00007100
0000C8 4770 F0D8           000D8    72          BNE   MNOSHIFT          BRANCH, IF NOT EQUAL            00007200
0000CC 5B30 F23C           0023C    73          S     R3,=F'1'          DECREMENT M3 TO SHIFT RIGHT     00007300
0000D0 4720 F0D8           000D8    74          BP    MNOSHIFT                                          00007400
0000D4 4130 0001           00001    75          LA    R3,1              M MUST BE .GE. 1                00007500
0000D8 4360 F210           00210    76 MNOSHIFT IC    R6,WORK3                                          00007600
0000DC 1460                         77          NR    R6,R0             R6 := EXPONENT FIELD            00007700
0000DE 1A63                         78          AR    R6,R3             M3 = MAX(M1,M2)                 00007800
0000E0 1960                         79          CR    R6,R0             EXP OVERFLOW DUE TO UNNORMALIZATION? 00007900
0000E2 4720 F1F8           001F8    80          BH    MOVFL             YES, GO SIGNAL EXP OVERFLOW     00008000
0000E6 4360 F210           00210    81          IC    R6,WORK3          R6 := SIGN + EXPONENT           00008100
0000EA 1A63                         82          AR    R6,R3                                             00008200
0000EC 0660                         83          BCTR  R6,0              DECREMENT M3                    00008300
0000EE 4260 F220           00220    84          STC   R6,DBZ                                            00008400
0000F2 6E00 F220           00220    85          AW    F0,DBZ            UNNORMALIZE BY (M-1)            00008500
0000F6 4260 F228           00228    86          STC   R6,MINCR                                          00008600
0000FA 6A00 F228           00228    87          AD    F0,MINCR          ROUND ON DIGIT TO BE SHIFTED OFF 00008700
0000FE 4166 0001           00001    88          LA    R6,1(R6)                                          00008800
000102 4260 F220           00220    89          STC   R6,DBZ                                            00008900
000106 6E00 F220           00220    90          AW    F0,DBZ            UNNORMALIZE THE PRODUCT         00009000
00010A 9816 D018           00018    91 MRETURN  LM    R1,R6,24(R13)     RESTORE REGISTERS               00009100
00010E 5811 0008           00008    92          L     R1,8(R1)          GET RETURN VALUE ADDRESS        00009200
000112 6001 0000           00000    93          STD   F0,0(R1)          MOVE RETURN VALUE TO TARGET     00009300
000116 07FE                         94          BR    14                RETURN                          00009400

                                    96 *    BOTH OPERANDS WERE PRECISE                                  00009600
                                    97 *    NORMALIZED RESULTANT FRACTION HAS AT MOST FOURTEEN HEX DIGITS 00009700
000118 2C02                         98 MPRECISE MDR   F0,F2             DO PRECISE MULTIPLY             00009800
00011A 2200                         99          LTDR  F0,F0             TEST FOR ZERO FRACTION          00009900
00011C 4780 F1D0           00100   100          BZ    MZEROFIX                                          00010000
000120 2822                        101          SDR   F2,F2             DO QUICK TEST FOR PRECISE RESULT 00010100
000122 2866                        102          SDR   F6,F6                                             00010200
000124 6841 0000           00000   103          LD    F4,0(R1)                                          00010300
000128 3824                        104          LER   F2,F4             COPY H.O. PART OF NUMBER        00010400
00012A 2924                        105          COR   F2,F4             IS L.O. PART OF NUMBER ZERO     00010500
00012C 4770 F13C           0013C   106          BNE   MLONGTST                                          00010600
000130 6842 0000           00000   107          LD    F4,0(R2)                                          00010700
000134 3864                        108          LER   F6,F4                                             00010800
000136 2964                        109          CDR   F6,F4             IS L.O. 8 DIGITS ZERO           00010900
000138 4780 F10A           0010A   110          BE    MRETURN           RETURN WITH PRECISE PRODUCT     00011000
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                  21 JUN 72

00013C                           112 MLONGTST EQU   *   LONG TEST CONSIST OF GET L.O. PART OF PRODUCT      00011200
                                 113 *                     FRACTION AND EXAMINE FOR NON-ZERO DIGITS      00011300
00013C 907B 0030          00030  114         STM   R7,R11,48(R13)     SAVE MORE REGISTERS               00011400
000140 9867 1000          00000  115         LM    R6,R7,0(R1)        PICK UP 1ST OPERAND IN R6,R7      00011500
000144 4166 0000          00000  116         LA    6,0(6)             REMOVE SIGN AND EXPONENT          00011600
000148 8060 0004          00004  117         SLDL  R6,4               DECOMPOSE IT INTO H.O. 7 DIGITS   00011700
00014C 8970 0004          00004  118         SRL   R7,4               AND L.O. 7 DIGITS                 00011800
000150 9889 2000          00000  119         LM    R8,R9,0(R2)                                          00011900
000154 4188 0000          00000  120         LA    8,0(8)             REMOVE SIGN AND EXPONENT          00012000
000158 8D80 0004          00004  121         SLDL  R8,4               SAME FOR 2ND OPERAND IN R8,R9     00012100
00015C 8890 0004          00004  122         SRL   R9,4                                               00012200
000160 1886                      123         LR    R11,R6                                             00012300
000162 1868                      124         LR    R6,R8                                              00012400
000164 1CA9                      125         MR    R10,R9             H.O. OPER1 * L.O. OPER2           00012500
000166 1C87                      126         MR    R8,R7              L.O. OPER1 * L.O. OPER2           00012600
000168 1C66                      127         MR    R6,R6              L.O. OPER1 * H.O. OPER2           00012700
00016A 8080 0004          00004  128         SLDL  R8,4               SUM UP THESE PRODUCTS WITH        00012800
00016E 1E87                      129         ALR   R8,R7              CORRECT SCALING AND IGNORE        00012900
000170 1E8B                      130         ALR   R9,R11             CARRY INTO H.O. PART OF PRODUCT   00013000
000172 8C80 0004          00004  131         SRDL  R8,4                                               00013100
000176 1288                      132         LTR   R8,R8              TEST FOR NOZERO DIGITS            00013200
000178 4770 F190          00190  133         BNZ   MUNNORM                                            00013300
00017C 1299                      134         LTR   R9,R9              TEST FOR NON-ZERO DIGITS          00013400
00017E 4770 F190          00190  135         BNZ   MUNNORM                                            00013500
000182 981B D018          00018  136         LM    R1,R11,24(R13)     RESTORE REGISTERS                 00013600
000186 5811 0008          00008  137         L     R1,8(R1)           GET RETURN VALUE ADDRESS          00013700
00018A 6001 0000          00000  138         STD   F0,0(R1)           MOVE RETURN VALUE TO TARGET       00013800
00018E 07FE                      139         BR    R14                RETURN WITH PRECISE PRODUCT       00013900

000190 6000 F210          00210  141 MUNNORM STD   F0,WORK3       MAKE FRACTION IMPRECISE WITH ROUNDING  00014100
000194 4330 F210          00210  142         IC    R3,WORK3           FETCH NORMALIZED EXPONENT         00014200
000198 4230 F228          00228  143         STC   R3,MINCR           FIX EXPONENT FOR ROUND UP         00014300
00019C 6A00 F228          00228  144         AD    F0,MINCR           ROUND UP                          00014400
0001A0 4133 0001          00001  145         LA    R3,1(R3)           UP EXPONENT TO SHIFT FRACTION RIGHT 00014500
0001A4 4230 F220          00220  146         STC   R3,D8Z             STORE EXPONENT FOR UNNORMALIZATION 00014600
0001A8 6E00 F220          00220  147         AW    F0,D8Z             UNNORMALIZE RESULT                00014700
0001AC 981B D018          00018  148         LM    R1,R11,24(R13)     RESTORE REGISTERS                 00014800
0001B0 5811 0008          00008  149         L     R1,8(R1)           GET RETURN VALUE ADDRESS          00014900
0001B4 6001 0000          00000  150         STD   F0,0(R1)           MOVE RETURN VALUE TO TARGET       00015000
0001B8 07FE                      151         BR    R14                RETURN WITH IMPRECISE PRODUCT     00015100

                                 153 *   HANDLE CASE OF ZERO FACTORS                                  00015300
0001BA 9500 1000          00000  154 MZPROD1 CLI   0(R1),X'00'        CHECK 1ST OPERAND FOR TRUE ZERO   00015400
0001BE 4780 F10A          0010A  155         BE    MRETURN                                            00015500
0001C2 9500 2000          00000  156 MZPROD2 CLI   0(R2),X'00'        CHECK 2ND OPERAND FOR TRUE ZERO   00015600
0001C6 4770 F1D0          00100  157         BNE   MZEROFIX           BRANCH, IF RELATIVE ZERO          00015700
0001CA 2B00                      158         SDR   F0,F0              PRODUCE TRUE ZERO IN F0           00015800
0001CC 47F0 F10A          0010A  159         B     MRETURN                                            00015900

000100 4100 007F          0007F  161 MZEROFIX LA    R0,X'7F'       SECTION TO PRODUCE RELATIVE ZERO RESULT 00016100
000104 4331 0000          00000  162         IC    R3,0(R1)           GET 1ST EXPONENT                  00016200
000108 1430                      163         NR    R3,R0                                              00016300
00010A 4342 0000          00000  164         IC    R4,0(R2)           GET 2ND EXPONENT                  00016400
00010E 1440                      165         NR    R4,R0                                              00016500
0001E0 1A34                      166         AR    R3,R4              ADD EXPONENTS                     00016600
```

```
  LOC  OBJECT CODE     ADDR1 ADDR2  STMT   SOURCE STATEMENT                                                21 JUN 72
0001E2 5B30 F238            00238   167          S     R3,=F'64'          CORRECT FOR EXCESS-64 ADD          00016700
0001E6 1930                         168          CR    R3,R0              IS RESULT EXPONENT GT 127          00016800
0001E8 4720 F1F8            001F8   169          BH    MOVFL                                                 00016900
0001EC 4230 F220            00220   170          STC   R3,DBZ             FIX UP EXPONENT OF RELATIVE ZERO   00017000
0001F0 6800 F220            00220   171          LD    F0,DBZ                                                00017100
0001F4 47F0 F10A            0010A   172          B     MRETURN            RETURN WITH RELATIVE ZERO          00017200

0001F8 6800 F230            00230   174 MOVFL    LD    F0,=0'1E+70'       SECTION TO SIGNAL EXP OVERFLOW     00017400
0001FC 6C00 F230            00230   175          MD    F0,=0'1E+70'                                          00017500

000200                              177 WORK1    DS    D                  WORK AREA FOR FIRST OPERAND        00017700
000208                              178 WORK2    DS    D                  WORK AREA FOR SECOND OPERNAD       00017800
000210                              179 WORK3    DS    D                  WORK AREA FOR RESULT               00017900
000218 0000000000000000             180 TZERO    DC    D'0.0'             FLOATING POINT ZERO # NOT CHANGED # 00018000
000220 0000000000000000             181 DBZ      DC    D'0.0'             ZERO FRACTION CONSTANT             00018100
000228 FF00000000C00008             182 MINCR    DC    X'FF00000000000008'    ROUNDING FRACTION CONSTANT     00018200
000230                              183          LTORG                                                       00018300
000230 7B172EBA060DC730             184                =D'1E+70'
000238 00000040                     185                =F'64'
00023C 00000001                     186                =F'1'
                                    187 *   REGISTER DEFINITIONS                                            00018400
000000                              188 R0       EQU   0                                                     00018500
000001                              189 R1       EQU   1                                                     00018600
000002                              190 R2       EQU   2                                                     00018700
000003                              191 R3       EQU   3                                                     00018800
000004                              192 R4       EQU   4                                                     00018900
000005                              193 R5       EQU   5                                                     00019000
000006                              194 R6       EQU   6                                                     00019100
000007                              195 R7       EQU   7                                                     00019200
000008                              196 R8       EQU   8                                                     00019300
000009                              197 R9       EQU   9                                                     00019400
00000A                              198 R10      EQU   10                                                    00019500
00000B                              199 R11      EQU   11                                                    00019600
00000D                              200 R13      EQU   13                                                    00019700
00000E                              201 R14      EQU   14                                                    00019800
00000F                              202 R15      EQU   15                                                    00019900
000000                              203 F0       EQU   0                                                     00020000
000002                              204 F2       EQU   2                                                     00020100
000004                              205 F4       EQU   4                                                     00020200
000006                              206 F6       EQU   6                                                     00020300
                                    207          END                                                         00020400
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                                21 JUN 72

                                     2 *   DCL DL2PIVP ENTRY ( FLOAT BIN(53),     /*   X(1)              */ 00000200
                                     3 *                      FLOAT BIN(53),     /*   Y(1)              */ 00000300
                                     4 *                      FIXED BIN(15) )    /*   LENGTH OF VECTORS  */ 00000400
                                     5 *                RETURNS ( FLOAT BIN(53) ); /*   INNER PRDUCT      */ 00000500
                                     6          PRINT NOGEN                                                  00000600
000000                               7 OL2PIVP  CSECT                                                       00000700
                                     8 *   REGISTER DEFINITIONS.                                            00000800
000002                               9 RX       EQU   2                  RX -> X(I)                         00000900
000003                              10 RY       EQU   3                  RY -> Y(I)                         00001000
000004                              11 RL       EQU   4                  RL := LENGTH OF VECTOR             00001100
000005                              12 RXD      EQU   5                  RXD := DISPLACEMENT OF X(I)'S      00001200
000006                              13 RYD      EQU   6                  RYD := DISPLACEMENT OF Y(I)'S      00001300
000007                              14 RMASK    EQU   7                  RMASK := X'7F'                     00001400
000008                              15 RMAXM    EQU   8                  RMAXM := MAX M                     00001500
000009                              16 RC14     EQU   9                  RC14 := 14                         00001600
00000A                              17 RA       EQU   10                 SCRATCH REGISTER                   00001700
00000B                              18 RB       EQU   11                 SCRATCH REGISTER                   00001800
000000                              19 FA       EQU   0                  FA := X(I) AND PRODUCT             00001900
000002                              20 FB       EQU   2                  FB := Y(I)                         00002000
000004                              21 FC       EQU   4                  FC := INNER PRODUCT                00002100
000006                              22 FD       EQU   6                                                     00002200
000008                              23 XD       EQU   8                  DOUBLE WORD VECTOR                 00002300
000008                              24 YD       EQU   8                  DOUBLE WORD VECTOR                 00002400
000000                              25          USING *,15                                                  00002500
000000 47F0 F00C            0000C   26          B     *+12                                                  00002600
000004 07                           27          DC    AL1(7)                                                00002700
000005 D6D3F207C9E5D7               28          DC    CL7'OL2PIVP'                                          00002800
00000C 90EB D00C            0000C   29          STM   14,11,12(13)                                          00002900
000010 4530 F084            00084   30          BAL   3,PIVP01                                              00003000
000014 C2800070                     31 PIVP00   DC    XL4'C2800070'      STATIC SAVE AREA                 00003100
000018 0000000000000000             32          DC    27F'0'                                                00003200
000034 1822                         33 PIVP01   SR    2,2                                                   00003300
000036 5023 0050            00050   34          ST    2,80(3)                                               00003400
00003A 4133 0000            00000   35          LA    3,0(3)                                                00003500
00003E 58F0 F290            00290   36          L     15,=V(OL2PROLG)    ESTABLISH PL/1 ENVIRONMENT         00003600
000042 05EF                         37          BALR  14,15                                                 00003700
                                    38          DROP  15                                                    00003800
000014                              39          USING PIVP00,13          BASE REG = R13                     00003900
000044 9825 1000            00000   40          LM    2,5,0(1)                                              00004000
000048 4844 0000            00000   41          LH    RL,0(4)            RL := LENGTH OF VECTORS            00004100
00004C 4155 0000            00000   42          LA    5,0(5)                                                00004200
000050 5050 D234            00248   43          ST    5,ZADR             SAVE ADDRESS OF TARGET             00004300
000054 2844                         44          SDR   FC,FC                                                 00004400
000056 6040 D224            00238   45          STD   FC,SUM             INITIALIZE SUM                     00004500
00005A 1888                         46          SR    RMAXM,RMAXM        RMAXM := MAX M                     00004600
00005C 4150 0008            00008   47          LA    RXD,XD             RXD := DISPLACEMENT WITH X(I)'S    00004700
000060 4160 0008            00008   48          LA    RYD,YD             RYD := DISPLACEMENT WITH Y(I)'S    00004800
000064 4190 000E            0000E   49          LA    RC14,14            RC14 := CONSTANT 14                00004900
000068 4170 007F            0007F   50          LA    RMASK,X'7F'        RMASK := MASK FOR EXPONENT FIELD   00005000

00006C 9500 2000      00000         52 PRECLOOP CLI   0(RX),X'00'   PRECISE INNER PRODUCT SECTION          00005200
000070 4780 00FC            00110   53          BE    NEXTITEM           BRANCH, IF X(I) IS TRUE ZERO       00005300
000074 91F0 2001      00001         54          TM    1(RX),X'F0'                                           00005400
000078 4780 D128            0013C   55          BZ    IMPX               BRANCH, IF X(I) IS IMPRECISE       00005500
00007C 9500 3000      00000         56          CLI   0(RY),X'00'                                           00005600
```

```
  LOC  OBJECT CODE     ADDR1 ADDR2  STMT   SOURCE STATEMENT                                  21 JUN 72
000000 4780 00FC            00110    57           BE    NEXTITEM          BRANCH, IF Y(I) IS TRUE ZERO      00005700
000004 91F0 3001      00001         58           TM    1(RY),X'F0'                         00005800
000008 4780 0128            0013C   59           BZ    IMPY              BRANCH, IF Y(I) IS IMPRECISE      00005900
00000C 5020 025C            00270   60           ST    RX,MPARM                            00006000
000010 5030 0260            00274   61           ST    RY,MPARM+4                          00006100
000014 4110 025C            00270   62           LA    1,MPARM                             00006200
000018 58F0 0254            00268   63           L     15,VMPY           CALL PRECISE MPY ROUTINE          00006300
00001C 05EF                         64           BALR  14,15                               00006400
00001E 91F0 021D      00231         65           TM    MPROD+1,X'F0'                       00006500
000022 4780 0114            00128   66           BZ    IMPMPROD          BRANCH, IF PRODUCT WAS IMPRECISE  00006600
000026 4110 0268            0027C   67           LA    1,APARM                             00006700
00002A 58F0 0258            0026C   68           L     15,VADD           CALL PRECISE ADD ROUTINE          00006800
00002E 05EF                         69           BALR  14,15                               00006900
000030 9500 0224      00238         70           CLI   SUM,X'00'                           00007000
000034 4780 00FC            00110   71           BE    NEXTITEM .        BRANCH, IF SUM WAS TRUE ZERO      00007100
000038 91F0 0225      00239         72           TM    SUM+1,X'F0'                         00007200
00003C 4780 0120            00134   73           BZ    IMPSUM            BRANCH, IF SUM WAS IMPRECISE      00007300
000040 1A25                         74 NEXTITEM  AR    RX,RXD            POINT TO X(I+1)                   00007400
000042 1A36                         75           AR    RY,RYD            POINT TO Y(I+1)                   00007500
000044 4640 00A8            0008C   76           BCT   RL,PRECLOOP                         00007600
000048 6800 0224            00238   77           LD    0,SUM                               00007700
00004C 5810 0234            00248   78           L     1,ZADR                              00007800
000050 6001 0000            00000   79           STD   0,0(1)            MOVE PRECISE SUM TO TARGET        00007900
000054 47F0 020C            00220   80           B     EXIT                                00008000

000058 6800 021C            00230   82 IMPMPROD  LD    FA,MPROD                            00008200
00005C 6840 0224            00238   83           LD    FC,SUM                              00008300
000060 47F0 0180            001C4   84           B     IMPLPMP                             00008400
000064 6840 0224            00238   85 IMPSUM    LD    FC,SUM                              00008500
000068 47F0 0182            001C6   86           B     IMPLPSUM                            00008600
00006C                              87 IMPY      EQU   *                                   00008700
00006C 6840 0224            00238   88 IMPX      LD    FC,SUM                              00008800

000070 6802 0000            00000   90 IMPLOOP   LD    FA,0(RX)      IMPRECISE INNER PDUCT SECTION       00009000
000074 2200                         91           LTDR  FA,FA                               00009100
000076 4780 0182            001C6   92           BZ    NEXTONE           SKIP PRODUCT, IF ZERO FRACTION    00009200
00007A 43A2 0000            00000   93           IC    RA,0(RX)                            00009300
00007E 14A7                         94           NR    RA,RMASK          RA := EXP OF X(I)                 00009400
000080 19A9                         95           CR    RA,RC14           CHECK FOR POSSIBLE EXP UNDERFLOW  00009500
000082 4780 0152            00166   96           BNL   XDPOK                               00009600
000086 41AA 000E            0000E   97           LA    RA,14(RA)                           00009700
00008A 6000 022C            00240   98           STD   FA,DBWK                           00009800
00008E 42A0 022C            00240   99           STC   RA,DBWK           USE DIFFERENT EXPONENT FILED      00009900
000092 6800 022C            00240  100           LD    FA,DBWK                             00010000
000096 6A00 023C            00250  101 XDPOK     AD    FA,TZERO          NORMALIZE  TO DETERMINE M         00010100
00009A 6000 022C            00240  102           STD   FA,DBWK                           00010200
00009E 4380 022C            00240  103           IC    R8,DBWK                             00010300
0000A2 14B7                        104           NR    R8,RMASK          R8 := NORMALIZED EXPONENT         00010400
0000A4 1BA8                        105           SR    RA,R8             NOW RA HAS M OF X(I)              00010500
0000A6 19A8                        106           CR    RA,RMAXM                            00010600
0000A8 47D0 016A            0017E  107           BNH   *+6               WANT MAX M                        00010700
0000AC 18BA                        108           LR    RMAXM,RA                            00010800
0000AE 6823 0000            00000  109           LD    F8,0(RY)          GET Y(I)                          00010900
0000B2 2222                        110           LTDR  F8,F8                               00011000
0000B4 4780 0182            001C6  111           BZ    NEXTONE           SKIP PRODUCT, IF ZERO FRACTION    00011100
```

```
  LOC  OBJECT CODE     ADDR1 ADDR2  STMT   SOURCE STATEMENT                                  21 JUN 72

000188 43A3 0000            00000   112          IC    RA,D(RY)                                    00011200
00018C 14A7                         113          NR    RA,RMASK      RA := EXP OF Y(I)             00011300
00018E 19A9                         114          CR    RA,RC14       CHECK FOR POSSIBLE EXP UNDERFLOW00011400
000190 4780 0190            001A4   115          BNL   YOPOK                                       00011500
000194 41AA 000E            0000E   116          LA    RA,14(RA)                                   00011600
000198 6020 022C            00240   117          STD   F0,DBWK                                     00011700
00019C 42A0 022C            00240   118          STC   RA,DBWK       USE DIFFERENT EXPONENT FILED  00011800
0001A0 6820 022C            00240   119          LD    F8,DBWK                                     00011900
0001A4 6A20 023C            00250   120 YOPOK    AD    F8,TZERO      NORMALIZE  TO DETERMINE M    00012000
0001A8 6020 022C            00240   121          STD   F0,DBWK                                     00012100
0001AC 43B0 022C            00240   122          IC    RB,DBWK                                     00012200
0001B0 14B7                         123          NR    RB,RMASK      RB := NORMALIZED EXPONENT     00012300
0001B2 18AB                         124          SR    RA,RB         NOW RA HAS M OF Y(I)          00012400
0001B4 19A8                         125          CR    RA,RMAXM                                    00012500
0001B6 47D0 01A8            0018C   126          BNH   *+6           WANT MAX M                    00012600
0001BA 18BA                         127          LR    RMAXM,RA                                    00012700
0001BC 6802 0000            00000   128          LD    F0,0(RX)      GET  INPUT X(I)               00012800
0001C0 6C03 0000            00000   129          MD    F0,0(RY)      DO NORMALIZED MULTIPLY        00012900
0001C4                              130 IMPLPMP  EQU   *                                           00013000
0001C4 2A40                         131          ADR   F0,F0         ACCUMULATE SUM                00013100
0001C6                              132 IMPLPSUM EQU   *                                           00013200
0001C6 1A25                         133 NEXTONE  AR    RX,RXD        POINT TO X(I+1)               00013300
0001C8 1A36                         134          AR    RY,RYD        POINT TO Y(I+1)               00013400
0001CA 4640 012C            00140   135          BCT   RL,IMPLOOP                                  00013500

0001CE 1288                         137          LTR   RMAXM,RMAXM   M MUST BE POSITIVE           00013700
0001D0 4720 01C4            001D8   138          BP    *+8                                         00013800
0001D4 4180 0001            00001   139          LA    RMAXM,1                                     00013900
0001D8 1989                         140          CR    RMAXM,RC14                                  00014000
0001DA 4740 01CE            001E2   141          BL    *+8           M MUST BE LESS THAN 14        00014100
0001DE 4180 0000            00000   142          LA    RMAXM,13                                    00014200
0001E2 6040 022C            00240   143          STD   F0,DBWK                                     00014300
0001E6 43A0 022C            00240   144          IC    RA,DBWK                                     00014400
0001EA 14A7                         145          NR    RA,RMASK      RA := EXPONENT FIELD          00014500
0001EC 1AA8                         146          AR    RA,RMAXM                                    00014600
0001EE 19A7                         147          CR    RA,RMASK      EXP OVERFLOW DUE TO UNNORMALIZATION? 00014700
0001F0 4720 0212            00226   148          BH    VPOVFL        YES, GO SIGNAL OVERFLOW       00014800
0001F4 43A0 022C            00240   149          IC    RA,DBWK       GET NORMALIZED SIGN + EXPONENT00014900
0001F8 1AA8                         150          AR    RA,RMAXM      RA := (SIGN+ EXP) OF RESULT   00015000
0001FA 06A0                         151          BCTR  RA,0          DECREMENT M                   00015100
0001FC 42A0 0244            00258   152          STC   RA,DBZ                                      00015200
000200 6E40 D244            00258   153          AW    F0,DBZ        UNNORMALIZE BY (M-1)          00015300
000204 42A0 D24C            00260   154          STC   RA,MINCR                                    00015400
000208 6A40 024C            00260   155          AD    F0,MINCR      ROUND ON DIGIT TO BE SHIFTED OFF00015500
00020C 41AA 0001            00001   156          LA    RA,1(RA)                                    00015600
000210 42A0 D244            00258   157          STC   RA,DBZ                                      00015700
000214 6E40 0244            00258   158          AW    F0,DBZ        UNNORMALIZE THE SUM           00015800
000218 5810 0234            00248   159          L     1,ZADR                                      00015900
00021C 6041 00D0            00000   160          STD   F0,0(1)       MOVE IMPRECISE SUM TO TARGET  00016000

000220                              162 EXIT     EQU   *                                           00016200
000220 58F0 0280            00294   163          L     15,=V(DL2EPLOG)                             00016300
000224 07FF                         164          BR    15            RETURN                        00016400

000226 6860 0274            00288   166 VPOVFL   LD    F0,=D'1E+70'  SECTION TO SIGNAL EXP OVERFLOW 00016600
```

```
  LOC  OBJECT CODE     ADDR1 ADDR2  STMT   SOURCE STATEMENT                                                    21 JUN 72
000228 6060 0274            00288   167          MO    F0,=D'1E+70'                                                00016700

000230                              169 MPROO    OS    0                                                           00016900
000238                              170 SUM      DS    D                                                           00017000
000240                              171 OBWK     OS    0                    WORK AREA                              00017100
000248                              172 ZADR     OS    F                    ADDRESS OF TARGET                      00017200
00024C 00000000
000250 0000000000000000             173 TZERO    DC    0'0'                 CONSTANT TRUE ZERO                     00017300
000258 0000000000000000             174 DBZ      DC    D'0'                 ZERO FRACTION CONSTANT                 00017400
000260 0000000000000008             175 MINCR    DC    X'0000000000000008'  ROUNDING FRACTION CONSTANT            00017500
000268 00000000                     176 VMPY     DC    V(PIMPY)                                                    00017600
00026C 00000000                     177 VADD     DC    V(@OL2A00)                                                  00017700
000270                              178 MPARM    DS    A                    ADDRESS OF X(I)                        00017800
000274                              179          OS    A                    ADDRESS OF Y(I)                        00017900
000278 80000230                     180          DC    X'80',AL3(MPROO)                                            00018000
00027C 00000230                     181 APARM    DC    A(MPROO)                                                    00018100
000280 00000238                     182          DC    A(SUM)                                                      00018200
000284 80000238                     183          DC    X'80',AL3(SUM)                                              00018300
000288                              184          LTORG                                                             00018400
000288 78172EBA060DC730             185                =D'1E+70'
000290 00000000                     186                =V(OL2PROLG)
000294 00000000                     187                =V(OL2EPLOG)
                                    188          END                                                               00018500
```

```
LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                                21 JUN 72

                                    2          COPY  PIFUN
                                    3          MACRO                                                     00000100
                                    4          PIFUN &ENTRY                                              00000200
                                    5 .*  MACRO TO PRODUCE SIGNIFICANT DIGIT FUNCTION ROUTINES.          00000300
                                    6 .*   USING PL/I FUNCTION CALLING CONVENTIONS AND PL/I LIBRARY.     00000400
                                    7 .*  MACRO VARIABLE: ENTRY IS THE LAST 3 OR 4 LETTERS IN ENTRY      00000500
                                    8 .*                    POINT <IHEXXXX>.                          00000600
                                    9 .*  OCL @LXXXX   ENTRY ( FLOAT BIN(53) )    /* UNNORMALIZED ARGUMENT */ 00000700
                                   10 .*               RETURNS ( FLOAT BIN(53) ); /* UNNORMALIZED FUNCTION */ 00000800
                                   11 @L&ENTRY CSECT                                                    00000900
                                   12          OS    0H                                                 00001000
                                   13          USING *,15                                               00001100
                                   14          B     *+12                                               00001200
                                   15          DC    AL1(7)                                             00001300
                                   16          DC    CL7'@L&ENTRY'                                      00001400
                                   17 .*  MAKE THIS ROUTINE TRANSPARENT BETWEEN CALLER AND LIBRAY ROUTINE 00001500
                                   18          STM   14,6,&ENTRY.SA X    SAVE REGISTERS IN OUR SAVE AREA  00001600
                                   19          DROP  15                                                 00001700
                                   20          LR    2,15 X              USE R2 AS BASE REGISTER        00001800
                                   21          USING @L&ENTRY,2                                         00001900
                                   22          L     3,0(1) X            R3 -> INPUT ARGUEMENT          00002000
                                   23          L     6,4(1) X            R6 -> TARGET                   00002100
                                   24          LD    0,0(3) X            GET INPUT ARGUMENT             00002200
                                   25          LTDR  0,0                                                00002300
                                   26          BZ    &ENTRY.Z2 X         BRANCH, IF ZERO FRACTION       00002400
                                   27 .*  DETERMINE M: THE NUMBER OF LEADING ZEROS IN HEX FRACTION       00002500
                                   28 .* SYS1.PL1LIB DOES NOT ACCEPT UNNORMALIZED INPUT                 00002600
                                   29          LA    0,X'7F'                                            00002700
                                   30          IC    4,0(3)                                             00002800
                                   31          NR    4,0 X               R4 := E1                       00002900
                                   32          AD    0,&ENTRY.TZ X       NORMALIZE TO FIND M1           00003000
                                   33          STD   0,&ENTRY.WK X       NORMALIZED ARGUEMENT FOR LIB ROUTINE 00003100
                                   34          IC    5,&ENTRY.WK X       GET NORMALIZED EXPONENT        00003200
                                   35          NR    5,0 X               R5 := NORMALIZED EXPONENT      00003300
                                   36          SR    4,5 X               R4 := M1                       00003400
                                   37          BP    *+8                                                00003500
                                   38 &ENTRY.Z1 LA   4,1 X               M1 MUST BE POSITIVE            00003600
                                   39          LA    1,&ENTRY.PL X       POINT TO NEW PARM LIST         00003700
                                   40          L     15,V&ENTRY X        CALL LIBRARY ROUTINE           00003800
                                   41          BALR  14,15                                              00003900
                                   42          LA    0,X'7F'                                            00004000
                                   43          IC    5,&ENTRY.FN                                        00004100
                                   44          NR    5,0 X               R5 := EXPONENT                 00004200
                                   45 .*  ADJUSTMENT RULE : M2 = M1.                                    00004300
                                   46          AR    5,4                                                00004400
                                   47          CR    5,0 X          EXP OVERLOW DUE TO UNNORMALIZATION?  00004500
                                   48          BH    &ENTRY.OVFL X       YES, GO SIGNAL OVERFLOW        00004600
                                   49          LD    0,&ENTRY.FN X       GET NORMALIZED FUNCTION VALUE  00004700
                                   50          IC    5,&ENTRY.FN X       R5 := SIGN+EXPONENT            00004800
                                   51          AR    5,4                                                00004900
                                   52          BCTR  5,0                                                00005000
                                   53          STC   5,&ENTRY.ZF                                        00005100
                                   54          AW    0,&ENTRY.ZF X       UNNORMALIZE BY (M2-1)          00005200
                                   55          STC   5,&ENTRY.INCR                                      00005300
                                   56          AD    0,&ENTRY.INCR X     ROUND ON DIGIT TO BE SHIFTED OFF 00005400
```

```
LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                    21 JUN 72
                                   57          LA    5,1(5)                                00005500
                                   58          STC   5,&ENTRY.ZF                           00005600
                                   59          AW    0,&ENTRY.ZF X       UNNORMALIZE BY M2 00005700
                                   60 &ENTRY.EXIT  EQU *                                   00005800
                                   61          STD   0,0(6) X            MOVE F(X) TO TARGET 00005900
                                   62          LM    14,6,&ENTRY.SA X    RESTORE REGISTERS   00006000
                                   63          BR    14 X                RETURN TO CALLER    00006100
                                   64          SPACE 1                                       00006200
                                   65 &ENTRY.Z2  EQU * X          ZERO FRACTION ARGUEMENT SECTION 00006300
                                   66          STD   0,&ENTRY.WK                           00006400
                                   67          CLI   0(3),X'00'                            00006500
                                   68          BE    &ENTRY.Z1 X         BRANCH BACK, IF TRUE ZERO 00006600
                                   69          SR    4,4                                   00006700
                                   70          IC    4,0(3)                                00006800
                                   71          S     4,=F'64' X          R4 := RELATIVE ZERO EXP 00006900
                                   72          BP    *+8                                   00007000
                                   73          LO    0,&ENTRY.IRZ X      F(X) FOR NON-POSITIVE EXP 00007100
                                   74          B     &ENTRY.EXIT                           00007200
                                   75          SPACE 1                                     00007300
                                   76 &ENTRY.OVFL LD 0,=D'1E+70'                           00007400
                                   77          MD    0,=D'1E+70'                           00007500
                                   78          SPACE 1                                     00007600
                                   79 &ENTRY.ZF  DC  D'0' X              CONSTANT ZERO FRACTION 00007700
                                   80 &ENTRY.TZ  DC  D'0' X              CONSTANT TRUE ZERO  00007800
                                   81 &ENTRY.WK  DS  D X                 WORK AREA           00007900
                                   82 &ENTRY.FN  DS  D X                 NORMALIZED FUNCTION VALUE 00008000
                                   83 &ENTRY.INCR DC X'0000000000000008' ROUNDING CONSTANT   00008100
                                   84 &ENTRY.IONE DC X'4000000000000010' IMPRECISE ONE       00008200
                                   85 &ENTRY.IRZ  DC X'4000000000000000' IMPRECISE ZERO      00008300
                                   86 &ENTRY.SA  DC  9F'0' X             OUR SAVE AREA       00008400
                                   87 V&ENTRY    DC  V(IHE&ENTRY) X      LIBRARY ROUTINE ENTRY POINT 00008500
                                   88 &ENTRY.PL  DC  A(&ENTRY.WK) X      PARMETER LIST FOR LIB ROUTINE 00008600
                                   89          DC    X'80',AL3(&ENTRY.FN)                00008700
                                   90          LTORG                                       00008800
                                   91          DROP 2                                      00008900
                                   92          MEND                                        00009000
```

IOGRAPHIC DATA T	1. Report No. UIUCDCS-R-72-530	2.	3. Recipient's Accession No.
le and Subtitle MPLEMENTATION OF BASIC SOFTWARE FOR SIGNIFICANT DIGIT ARITHMETIC			5. Report Date June 1972 6.
thor(s) TEVEN SEE SUN LAI			8. Performing Organization Rept. No.
rforming Organization Name and Address ept. of Computer Science niversity of Illinois at Urbana-Champaign rbana, Illinois 61801			10. Project/Task/Work Unit No. 11. Contract/Grant No. US NSF-GJ-328
ponsoring Organization Name and Address ational Science Foundation ashington, D.C. 20550			13. Type of Report & Period Covered 14.

upplementary Notes

bstracts

This report is concerned with the implementation of significant igit arithmetic using the unnormalized arithmetic of Metropolis nd Ashenhurst. One of the primary objectives was to design the asic software modules for inclusion into the OL/2 array language; owever, these modules are written in assembly language and there- ore are adaptable to other software systems for the IBM 360/370 achines. A discussion of significance arithmetic, including the ontrivial problem of input/output, is presented. Examples are rovided in Appendix A.

ey Words and Document Analysis. 17a. Descriptors

Significance arithmetic, unnormalized significant digit arithmetic.

Identifiers/Open-Ended Terms

COSATI Field/Group

vailability Statement	19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages 77
nlimited	20. Security Class (This Page UNCLASSIFIED	22. Price